Version 4: February 2023.

Computer Technology Ltd. and the Modular One.

Contents.

1. Introduction and CTL company history.

Computer Technology (later Information Technology) was started in 1965 by Iann Barron together with a group of computer engineers from Elliott Automation. The immediate reason for the company's foundation was the announcement of the Digital Equipment PDP11. Iann Barron, recounting the story many years later, believed that this was an ideal computer for Elliotts to manufacture and said that he had 'secured the offer of the European licence'; however the management of Elliotts were not interested. While it might have been sensible for Elliotts as an established company to market the PDP11, this was not a practicable proposition for a startup and so, once he had left Elliott Automation, Iann Barron's aim was to create a more powerful computer which would address the same market as the PDP11 – this was to become known as the minicomputer market.

Financing was secured remarkably quickly, the two initial backers being Arnaud de Vitry, who had been the principal backer of Digital Equipment, and Robert Maxwell, who had a vision of computers as the future of publishing. Unfortunately, Maxwell proved a very uncertain source of funds, which led to many difficulties for Computer Technology Ltd.

The concept of the Modular One was created around a number of ideas:

- The use of integrated circuits to build a computer. In particular the use of fast emitter coupled logic to gain a performance advantage. Modular One was one of the first computers to use integrated circuits and possibly the first to use emitter coupled logic integrated circuits.
- The use of a high performance core store with a 1 microsecond cycle time.
- A 16 bit word – at that time the fashion was for 12 or 24 bits, which was strongly preferred by the British establishment (for example, at Elliotts).
- A standardised interface to connect all the modules of the computer, enabling multiple processors, modular storage and flexible peripheral configurations.
- Multiprogramming, with relocatable program and data spaces and some degree of memory protection.
- Program and peripheral priority systems, with fast program switching.
- Multiprogramming operating system.

Work on the design was started in August 1965 in a terraced house in Luton. The first computer was delivered on schedule about three years later, to a company in Fenchurch Street, London. By this time Computer Technology Ltd. was operating from a Norman Foster designed factory in Hemel Hempstead. The Modular One met all Iann Barron's initial design objectives.

The main application areas were universities for research projects, the National Health Service, the MRC, CERN, instrumentation automation such as NMRs and a variety of advanced research applications, particularly multiple processors. More information on deliveries and dates is given in Section 2 below.

When he started the company, Iann Barron was helped by Tom Margerison (the founder of the *New Scientist*) who subsequently became Chairman of Computer Technology Ltd. Margerison changed the name to Information Technology. Iann Barron recalls that the company never had the money to develop a successor to the Modular One, so the computers gradually became non competitive and the company declined. Iann Barron left the company in 1973. It was eventually bought by ACT (Apricot) in January 1990 which was then acquired by Mitsubishi.

At the time of writing (2018) two Modular One computers are still known to exist, though neither is in regular working order. One system is in storage at the Museum of Science and Industry in Manchester. The other, an ex-NPL system used on the NPL packet switching network and the Scrapbook project (see video links here http://www.npl.co.uk/mathematics-scientific-computing/history-of-computing/), was bought in 1980 by Patrick Sugrue together with two teletypes and a high speed paper tape reader. It was used as a home computer up to 1982. The current plan is to restore this machine – see: http://redhawksys.com/index.htm

2. CTL Modular One deliveries and anecdotes about the sites.

Much of this delivery list has been compiled by Mike Gibbons evandmike@hotmail.com . After working at Elliott Brothers at Borehamwood from 1962 to 1968, Mike Gibbons joined Computer Technology Ltd. at Hemel Hempstead in 1968. He was an Installation and Commissioning Engineer until 1971, when he moved to the Systems Test Department. In 1974 he set up a new group called TEAM (Test Engineering And Methods) to introduce new products from design into manufacturing. Mike left CTL in 1992.

The tabular list of deliveries given below is only approximate, since the official company records have not yet come to light. Additionally, Reference [2.1] gives 37 more deliveries without dates. The entries in the Table are all for Modular One computers, of which well over a thousand are believed to have been delivered. The CTL Satellite One computer is not included in the list. This machine was a basic standard Modular One but with more or less a set configuration which included a tape reader and tape punch and Data Products line printer and minimum of one VDU. The Satellite One was very popular and sold in large numbers throughout the UK after 1972.

Background notes on some of the installations are given after the Table.

Delivery date	***No. of Systems***	***Customer/location***	***Comments***
July 1968	1	International Data Highways (IDH), Clerkenwell Rd. and Finsbury Square, London EC2	See Note 1. More IDH installations followed.
Oct 1968	1	Cambridge University Language Research Unit	Margaret Masterman
Nov 1968	1	Manchester University Dept of Psychology	Arthur Reader
1968/69	1	Oxford University Department of Mathematics	
1969	1	Oxford University, Science Labs (?)	
1969	1 + ?	Warwick University Dept of Computer Science	Colin Whitby-Strevens. See Note 2a.
1969	2	Cambridge University Mathematical Laboratory	Roger Needham. See Note 2b.
1969	1	Leeds University	
1969	1	Essex University	
1969	1	Sussex University Department of Mathematics	
1969	1	Autonomics Ltd.	
1969	1	Dundee Royal Infirmary	See Note 3.
1969?	1	MRC Laboratory, New Addenbrookes' Hospital, Cambridge.	
1969	1	Bristol University Medical School	See Note 4.
1969	1	Durham University School of Engineering Science	See Note 5.
1969	1	Dental practice in London	See Note 6.
1969	1 (later moved)	Medical Research Council, London	See Note 7.
1969	1 (later moved)	Medical Research Council, Manchester	See Note 7.
1969	1	Atomic Weapons Research Establishment, Aldermaston	
1969	1	UK Atomic Energy Authority, Culham	
1969 & 1970	2 + ?	International Data Highways, Finsbury Square	See Note 1.
1969/70	1	Oxford University Dept of English Language	
1970	1 + ?	National Physical Laboratory, Teddington	See Note 8.
1970	1 (later moved)	Medical Research Council, London	See Note 7.
1970	1 (later moved)	Medical Research Council, Manchester	See Note 7.
1970	1	John Radcliffe Hospital, Oxford	
1970	1	Aberystwyth University	
1970	1	Royal Victoria Hospital Belfast	
Before 1971	1 + ?	RAF Marham, Norfolk	See Notes 9.

Before 1971	1 + ?	RAF Special Projects (Nimrods).	See Notes 10.
Before 1971	1	MOD Royal Military Police, Euston Rd., London.	See Note 11.
1971	1 (later moved)	Medical Research Council, London	See Note 7.
1971	1	Sheffield Hallam University	
Oct 1971	1	Computing Science, Glasgow University	
Post 1971	1	Royal Navy submarine	See Notes 12.
Post 1971	1	European Space Agency, Holland	
May 1972	1	Edinburgh Royal Infirmary	See Note 7.
Post 1972	1	University College Hospital, London	
Post 1972	1	Aberdeen Hospital	
?	1	Dept. of Health & Social Security (DHSS) Blackpool	See Note 13.
?	1	DHSS Long Benton, North Tyneside	See Note 13.
?	440	DHSS offices throughout the UK	See Note 13
?	1	Supreme Headquarters Allied Powers Europe (SHAPE) Headquarters, Stanmore, Middlesex	
1971 - 74	several	The European Space Research and Technology Centre (ESTEC), Noordwijk, the Netherlands.	See Note 14.
Post 1974	many	Metropolitan Police Project 880	See Note 15
1972 onwards	many	ICL Kidsgrove	See Note 16
1970 – 75	Many	CERN, Geneva	See Note 17
1970 – 75	1	Computer Aided Design Centre, Cambridge.	See Note 20
1975	1	METEOSAT	See Note 18
1976	1	GEOSTAT	See Note 19

Additionally, Reference [2.1] gives 37 more Modular One deliveries but without dates. These, in the order quoted in Reference 2.1, are:

Conservatoire National des Arts et Metiers, Paris
Various Gas Boards (six Boards in all)
Southampton University
Science Research Council
Smiths industries
UKAEA
Loughborough University
Rutherford High Energy Lab
Brunel University
Cable & Wireless Ltd
Plessey Radar
Royal Radar Establishment
Imperial College
P L Hunter
Joseph Lucas Ltd
Signal Research & Development Establishment
Institute of Computer Science

Reading University
Cranfield Institute of Technology
Haden Carrier Ltd
Central Electricity Generating Board
Edinburgh Regional Computing Centre
Maryfield Hospital Dundee
Westfield College
The London Hospital
Department of Trade & Industry
Hawker Siddeley Aviation Ltd
Liverpool University
Ecodata (Cable & Wireless)
Strathclyde University
The City University
Surrey University

Notes on the Table.
1. The first system, delivered to Clerkenwell Road, was considered to be a CTL Beta site. Installed in dedicated air conditioned computer room in the basement of the building, followed by at least two more Modular Ones in subsequent years. The initial system was small with around four modules but then later expanded with more memory. The main system was eventually very large and was fitted with the new Burroughs B5500 1Mbyte fast disc, approximately 36" diameter spinning on the vertical plane with one head per track.

2(a). One of the first systems to have a Universal Interface which allowed users to design and implement their own interface. This Warwick system was connected to an NCR-Elliott 4120 in the Computer Room on the floor immediately above the Modular One. The fact that Mike Gibbons had worked on 4120 and 4130 computers at Elliotts was of some help in getting the connection working correctly. Mike remembers that Warwick was of the friendliest sites to visit.
Andrew Herbert adds: When I saw at least one Modular One system at Warwick University systems in late 1974 they were used by Colin Whitby-Strevens to support operating systems, distributed systems and computer network research. As I recall his group mostly programmed in BCPL and wrote their own operating system for the Modular One.

2(b). Andrew Herbert describes the Cambridge activity as follows. In 1975 there were two Modular Ones in use. The larger was principally used to support a student FORTRAN Teaching System on behalf of the University Computing Service; the smaller was used as a front-end processor for the CAP research computer. (The CAP project on memory protection ran from 1970 to 1977; see https://en.wikipedia.org/wiki/CAP_computer).

The FORTRAN Teaching System machine had a fixed disc used for swapping, an exchangeable disc for the user file system, a line printer, paper tape reader and punch and operator's console (an ASR33 teleprinter). In a separate building there was a classroom room with a number (20?) of KSR33 teletypes connected to the Modular One….This machine also acted as a line printer and paper tape punch

outstation for the IBM 370/165 serving the academic side of the Computer Laboratory.

The CAP front-end processor was a smaller machine physically but also with fixed disc, exchangeable disc, line printer, paper tape reader and punch, operator's console and a multiplexor for up to four KSR33 teletypes. It also had a custom hardware interface to the CAP in the form of a bi-directional parallel channel.

On both machines the fixed disc was a rebadged Burroughs device, 500K (or 1Mbyte?) 32 bit words capacity, and used for virtual memory swapping. The disc had one vane and 200 heads. The exchangeable disc, was a rebadged CDC device, capacity 7,000K 32 bit words, and held a user file system. The discs had 11 vanes with 20 heads. Each vane contained 200 cylinders, each comprised of 20 tracks. Each track was divided into 14 pages of 256 16 bit words. The heads were operated hydraulically.

The CAP Modular One had 16K of 16 bit word store. The FORTRAN system had a larger memory and occupied more cabinets than the CAP Modular One. The CAP Modular One ran the E2 Executive. The FORTRAN Teaching System ran the more powerful E4 Executive, a multi-tasking operating system…. Once or twice a year we would back up both systems by taking their exchangeable discs from the Computer Laboratory to another Modular One installation at the MRC Laboratory on the New Addenbrookes' Hospital site about 2 miles away.

The CAP front-end Modular One was decommissioned around 1978 when CAP was converted to use the then newly invented Cambridge Ring Local Area Network to access file, print and terminal servers. If I remember correctly the FORTRAN Teaching System was still in service when I left the Cambridge Computer Lab in 1985.

3. This system was housed in the Medical Research Laboratory, which contained numerous jars with various body parts immersed in fluid. The computer was actually first
delivered and installed at Edinburgh University for a day's demonstration to potential Northern and Scottish customers. It was then stripped down and re-packed for delivery and installation in Dundee. The whole process spanned five days, involving departure from Hemel Hempstead early on a Monday morning in a hired Transit van, Tuesday's Sales Demonstration, Wednesday's transport and re-installation at Dundee, Thursday's Acceptance Trials and Friday's confirmation meeting with satisfied users and return to Hemel Hempstead. Mike Gibbons and Chris Purkis were responsible for this demanding sequence.

4. This site was involved with medical training and research. The research, under Tom Williams, included the investigation of the brains of live cats.

5. This site was the first Modular One installation to have Analogue-to-Digital/Digital-to- Analogue converters attached.

6. This large private Dental Practice was close to Harley Street. The Modular One system was installed in the front window, in plain view from the street. It was one of

the first CTL systems to be delivered with a paper tape punch unit manufactured by the Tally Corporation.

7. This system was housed in a new Computer Building attached to the Hospital. It was the largest and most complex system ever installed by CTL. It re-used all the MRC Modular Ones previously installed in MRC sites in London and Manchester plus some new systems from the factory at Hemel Hempstead. The final MRC Edinburgh site included at least eight processor modules (type 1.11) interconnected, multiple storage consisting of type 1.21 core store and type 1.22 semiconductor memory. The peripheral equipment included type 1.32 GNT Tape Readers, type 1.33 Tally Punches, type 1.52 Disk Controllers with multiple 80 Mbyte, 300Mbyte single and double stacked disk drives (both hydraulic and voice coil actuated), Data Products Line Printers, CAD / ADC Controllers and type 1.09 9-Track Magnetic Tape Drives.

The main purpose of the MRC's Edinburgh Modular One system was to carry out testing of amniotic fluid taken from wombs of pregnant women throughout UK. The intention was to replace a manual process involving a number of women in a laboratory testing samples under a microscope and looking for errors/faults/defects in chromosomes. Sadly the computer system never worked satisfactorily and it was later used for Medical Research, Accounts and Patient Records. On a more positive note, more NHS installations were to come. Indeed, the Modular One became for a time the standard NHS laboratory computer.

The huge MRC Edinburgh complex was installed by Mike Gibbons (Lead Engineer) and Alfie Best. They remember that the CTL lorry driver Alec did a daily round trip from Hemel Hempstead to Edinburgh for several days, delivering equipment; on each journey northwards the 7 ton Ford lorry was loaded to the roof.

Mike and Alfie remember that the Edinburgh computer room was brand new and equipped with smoke and gas sensors. "One day while working under the false floor installing cabling, a serious smell became evident which set off the fire alarms. We left the room and sat outside, whilst several Fire Engines appeared from all over Edinburgh. The firemen soon identified the Computer Room as the location of the alarm, so we were promptly accused of smoking in the Computer Room! Having pointed out that we were not smoking and that there was still a serious smell in the room, the firemen made a further check and declared that the drain cover for the main sewer had lifted due to several bits of building debris becoming lodged in the pipe".

8. The Modular One at NPL was primarily used for used for software research, particularly for the *Scrapbook* project on information storage, retrieval and sharing – see https://www.youtube.com/watch?v=QqB0w1FkR3o *Scrapbook* involved a messaging system which, in due course, used NPL's pioneering 1.5 Mbps Packet Switching network – see: https://www.youtube.com/watch?v=tT4AaelwvV4

After NPL had finished with its Modular One, the computer was acquired in 1980 by Patrick Sugrue of Redhawk Systems, Dorchester, Dorset. Patrick has been restoring the machine to working order – see:
http://redhawksys.com/index_files/Page627.htm

9. Mike Gibbons remembers that systems for Marham were modified by RAF Engineers so that all contacts and the complete core of the core store were coated in oil supplied and used by RAF. Any equipment sent to this site was not re-usable by CTL so, if returned to Hemel Hempstead, any spares were scrapped.

10. The system was installed in special racking as specified by the RAF and delivered to an RAF site for installation in NIMROD aircraft. The computer(s) were maintained by RAF personnel by bulk part exchanges, but later serviced by CTL engineers due to the cost of repairs and high failure rates under service conditions.

11. This site was situated in Euston Road, London, protected by armed Military Police. The Modular One was used at the time of Irish (IRA) crisis. It was one of the first CTL installations to have 9 track reel to reel tape drives installed. Mike Gibbons remembers that he was subjected to a full security check of all his equipment and tools whenever he entered the building, due to the high security situation. “The door of the computer room was 4-inch solid oak. You were escorted everywhere. If you wanted to go to the lavatory you had to leave the toilet door open with the guard standing near you”.

12. The Modular One system was modified to fit into special racking for installation in a Royal Navy submarine.

13. An initial system was delivered to both the Blackpool and Long Benton sites, for an assessment of central records for DHSS. These were followed by a few more systems at both sites. Based on this experience, a special DHSS Modular One computer was designed around TTL 2901 bit- sliced processors. Mike Gibbons was the project leader. He remembers that, during manufacture, bar codes were used for the first time for part/serial numbers. An IBM PC with custom-built control box was used as a central station, enabling a senior wireman (Chris Silver) to assemble the system and use the PC to control the tests and issue instructions. This led to lower labour costs at CTL and improved fault finding. Once manufactured, one of these special Modular One systems went to each of the 440 DHSS offices throughout the UK.

14. A number of Modular One systems were delivered to ESTEC in Holland, who then shipped them to the Guiana Space Centre at Kourou in French Guiana on the north Atlantic coast of South America. At Kourou, satellites were launched using the French Ariane Rocket as the launch vehicle.

15. Mike Gibbons remembers that this was an £880K project which involved the delivery of a large number of systems to the Metropolitan Police. One substantial Modular One system was installed at Scotland Yard, housed in a lead lined room on the same floor as the Flying Squad, sometimes called the Central Robbery Squad or *The Sweeney*.

16. Numerous Modular One systems were delivered to ICL Kidsgrove over a period, at the rate of at least five systems every fortnight. They were specially configured for ICL, painted in ICL colours and used as Front-End Processors to both 1900 and 2900 mainframes.

They were initially badged as ICL 7905s. At some point in the late 1970s, a revised version of the hardware was released by CTL (this may or may not have been a cover story to disguise a price change) and assumed the 7906 designation. At the same time another version allowing much less user flexibility was announced at the 7904. This was intended to cover the same ground as the (ICL-designed) 7903 aka PF56 which was by then at the end of its sales life. As front-end communications processors, the Modular Ones in some cases allowed in excess of 100 terminals to be connected to an ICL mainframe system. Delivery to end-users and maintenance was carried out by ICL personnel.

17. At least 15 Modular One systems were delivered to CERN as part of their Scientific Programme, between 1970 and 1975. All systems were of the same configuration and included Data Products 600 LPM Line printers. Many (all?) of the CTL computers were used as Remote Input Output stations for CERN's CDC 7600 computer.

18. A Modular One system in custom racking was delivered to UMETSAT (European Organisation for the Exploitation of Meteorological Satellites) in Darmstadt. EUMETSAT operates the Meteosat series of geostationary weather satellites.

19. GEOSAT (GEOdetic SATellite) was a US Navy programme for oceanographic observations. The first GEOSAT satellite was launched in 1985. It is not known what part a Modular One system played in the oceanographic programme, or where the computer was sited – though it was first delivered to Holland (ESTEC).

20. Other installations for which details are sparse.
(a). A CTL Modular One computer with a half-inch magnetic tape system was connected to the CAD Centre's Atlas 2 computer at Madingley Road Cambridge in the early 1970s.

(b). The Department of Engineering Mathematics at Queens University, Belfast, had a Modular One running an interactive system with dumb terminals.

(c). There may have been a Modular One in Dublin at the government's Central Data Processing Services (CDPS).

Information believed to be correct as of March 2019.

Reference.
2.1. *Modular One.* 25-page glossy brochure with art-work. Publication date not given but deduce it was 1971 or 1972.

3. CTL Modular One computer: Systems Architecture and Instruction set.

Contents. *Page*

3.1. Systems architecture.

3.1.1. Overall system configuration.

A Modular One system consists of a combination of processor units, store units and peripherals. These units can be 'freely combined' although each store is only generally accessible to one processor unless a special store switching system is introduced. Besides the usual Arithmetic Unit and program-accessible central registers, the Processor cabinet contains a Communications Multiplexor and space for up to 24K of 16-bit words of integral fast (0.75 microsecond cycle time) core store. Later, a semiconductor store was made available. Each processor is in fact capable of addressing 56K words of direct store, which consists of 24K words of integral store and the remainder (if required) being made up of one or more external

store modules. The total range of addresses is actually divided between those referring to read/write memory locations and those referring to peripheral devices (eg Input/Output equipment). See section 3.2.5 below for more addressing details. All the (external) store modules operate independently, allowing accesses to be interleaved.

Computer Technology Ltd. (CTL) produced many versions of processors, storage modules and peripheral equipments during the life of the conpany. The notes below refer mainly to the standard type 1.14 processor. This operates from a single phase 240 V AC ring main and consumes a maximum of 1 KVA when all options are included.

A basic Modular One system with a paper tape reader and an ASR33 teletype for simple I/O.

Much larger Modular One configurations existed, as may be judged from the notes on *Deliveries and Applications* in section 2 of the Modular One description.

A model of a large Modular One installation.

Amongst the additional equipment available for large Modular One installations are the following [Reference 3.4]:

Paper tape readers of speeds up to 1,000 characters/sec.
Paper tape punches with speeds up to 120 characters/sec.
80-column card readers of speeds up to 1,000 cards/minute,
Lineprinters, either 80-column or 120-column printing, either 64 or 96 character set, with speeds up to 1,000 lines/minute,
Fast-access fixed discs, capacity up to 256Kwords, average access time of 8.45 millisec.
Exchangeable disc stores, capacity 14Mwords per pack, average access time of 13.6 millisecs
Magnetic tape transports,7-track or 9-track, with transfer speeds of up to 36 Kcharacters/ssec.

3.1.2. The Processor Interface.

Connection between modules in a Modular One system is via the Standard Modular One System Interface, which physically consists of two connectors, each with 33 pins. The type 1.14 Modular One processor has nine ports: one dedicated peripheral port, one dedicated to integral store, and seven codable for store or peripheral channels.

Incoming demands to a particular processor may be either hesitations or interrupts. When answering these the processor is said to act in *slave mode*. When the processor itself initiates an interface transfer it is said to act in *master mode*. Processor-initiated demands are routed to the correct store or peripheral channel by means of a channel address. Incoming demands are detected by means of a scan which samples the peripheral channels in priority order. Global inhibition of either interrupts or hesitations, or both, may be set by use of a *special state Instruction* (see section 3 below). A similar instruction is available for the selective rejection of individual interrupts. Both *special* and *normal* state code (provided that it does not violate its segment bounds) can set channel lockout. The Modular One concepts of *nrmal state* and *special state* are defined more fully in section 3.3.

In summary, each processor in a Modular One system can be in one of three modes when communicating with other modules:
Master mode: direct reference to other modules is allowed by program instructions.
Slave in Interrupt mode: interrupts allowed from other modules.
Slave in Hesitation mode: transfers allowed between peripherals and direct store.

3.1.3. Executive.

The type 1.14 processor is designed to operate with an Executive program, which contains facilities for the control of peripheral devices, for organising and allocating storage and processing time, for inter-program communications, for switching from one program to another and for dealing with the power on/off interrupts and error conditions. The Executive also contains service routines, accessible by software interrupts, which enhance the hardware facilities. See

section 1.3 below for more. A user-program in *Normal Mode* may use Function 30 to make a call to the Executive – see Appendix 3.1.7.

The X, Y and Z memory segmentation registers, together with the two execution states (*Normal State* and the non-interruptible privileged *Special State*) mean that the Executive acts as a self-protecting Operating System kernel.

3.1.4. Operator's switches.

Each procesor in a Modular One system has six On/Off switches: Power, Remote, On-line, Multiplexer, Operational, Load.

3.2.Registers, the Instruction set and timings.

3.2.1. Number representation.

The Modular One processor uses 16-bit words, which can represent either signed or unsigned bnary integers. The range of signed integers is thus -32,768 to + 32,767 and the range of unsigned numbers is 0 to 65,535. All addresses should be regarded as 16 bit positive integers, to word boundaries.

Double-length numbers occupy two 16-bit words, giving a range of -1,073,774,592 to + 1,073,741,823. The double length accumulator, [B , A] , has register B as its more significant half and register A as its less significant half, and holds an integer of value $B * 2^{15} + A$. The *normal* form of a double length integer, for use in arithmetic operations, is with the most-significant digit of A equal to zero.

3.2.2. Central registers.

There are eight principal (programmer-accessible) registers:

A	16 bits, main accumulator
B	16 bits, index register: auxiliary accumulator
M	16 bits, index register with indirect addressing: auxiliary accumulator
P	13 bits, program pointer
W	16 bits, local workspace stack pointer.
X	16-bit relocation register. Holds a pair of 8-bit values (lower and upper limits).
Y	16-bit relocation register. Holds a pair of 8-bit values (lower and upper limits).
Z	16-bit relocation register, Holds a pair of 8-bit values (lower and upper limits).

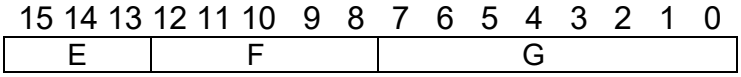
Registers B and A together form a 32-bit double-length accumulator [B,A]. The interpretation of address-information in registers W, X, Y and Z is given below. Other registers such as overflow indicators will be described later. The symbol Q in the Modular One manuals is used to denote an operand – see below under *Addressing modes*. Other symbols such as E, F, G, H, J, in the manuals do not denote physical registers. Rather, they are used to describe sub-fields in registers holding instructions, addresses, etc.

During arithmetic operations, registers A, B and M normally hold 16 bit signed numbers. They are provided with overflow indicators; Aovr , Bovr , and Movr respectively (though Movr cannot occur when a special state program is running).

Values held in A, B and M can each be tested for being less that zero, equal to zero, greater than zero, or arithmetic overflow. The B overflow indicator is set if the result of a multiply or shift lies outside the double length arithmetic range, or if the result of a divide overflows. Both the 16 bit W register and 13 bit P register hold positive numbers. Neither is provided with an overflow indicator.

3.2.3. Instruction format and addressing modes.

Instructions are 16 bits long, arranged as:



E 3 bits primary address mode. Specifies which segment X, Y or Z is to be accessed and how the base address is to be constructed.
Segments are described in section 3.2.6.
F 5 bits function (ie op code).
G 8 bits offset. This is used either a a Literal or in the construction of the store address by adding it to the base address.

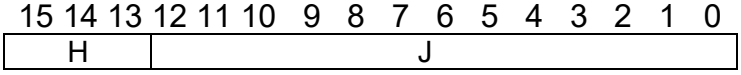
The E bits are interpreted differently, depending upon whether the processor is in *Normal Sate* or *Special State.* These program states, which relate to the handling of interrupts, are further described in section 3.3. The E bit possibilities are:

E value	***Addressing mode***	***Address construction, normal state***	***Addresss construction, special state***
0	Literal	G	G
1	Direct	P + G	P + G
2	Direct	G	G
3	Indirect	G	P + G
4	Direct	B + G	B + G
5	Direct	B + G	B + G
6	Direct	W + G	W + G
7	Indirect	W + G	W + G

3.2.4. Direct and Indirect Addressing.

Except for the *literal* mode, in which G is itself used as the operand for the function, a *primary* address is constructed from the contents of various registers as defined by the mode, with G as an offset. In the case of *direct* modes this address is that of the operand for the function. With *indirect* modes the primary address is used to fetch a secondary word from store, which then defines the operand address.

The format of the secondary word for *indirect* addressing is:



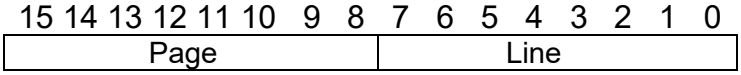
H 3 bits secondary address mode
J 13 bits base address (for data).

A secondary address is constructed from the contents of various registers as defined by the mode, with J as a data base address. This secondary address is always that of the operand for the function, i.e. there is only single-stage indirect addressing.

H value	*Address construction, normal state*	*Addresss construction, special state*
0	M + J	M + J
1	M + J	M + J
2	J	J
3	J	P +- J
4	W + J	M + J
5	W + J	M + J
6	B + J	B + J
7	B + J	B + J

3.2.5. Further addressing details: segments, context and peripherals.

The processor refers to both storage and peripherals by 16 bit addresses. These are considered, in address arithmetic, to be positive integers. They are more often written by the programmer as `<p, l>` where `p` is the page and `l` the line number, defined as follows:

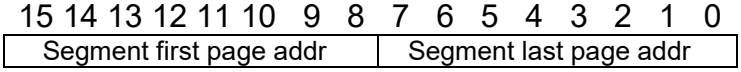


There are thus 256 pages in the complete address range, each of 256 lines. Addresses in the range <0, 0> to <223, 255> (ie numerically 0 to 57,343) are taken to refer to main memory (ie direct storage), and those in the range <224, 0> to <255, 255> (ie numerically 57,344 to 65,535) to peripheral devices.

3.2.6. Program Segments.

Every normal state program is allocated three segments, X, Y and Z. Each of these defines an address range to be used by the program, which may include direct storage or peripheral devices. The X (program) segment holds instructions and constants, and any direct store in it is *read only*. The Y (data) segment is used as a data area and as local workspace. Local workspace is the area in the Y segment to which W points – ie W acts as a *stack pointer.* The Z (file) segment is generally used to communicate with other programs and peripherals.

The position and extent of each segment is defined by a segment relocation register (the X, Y or Z register), which has the format:



Various address modes are provided to allow access to these segments. Also normal state code is fetched from store addresses that lie in the current X segment. All such addressing is relative to the appropriate segment first page address, and the hardware checks that the actual address constructed lies within the address range allocated to that segment. If the protection system detects an unacceptable address it generates a violation interrupt.

The settings of the X, Y and Z registers are changed via function 29 (see Appendix A3.1.6).

3.2.7. Program Contexts.

The condition of a normal state program at any given time, is characterised by its *working context:* { [M_{ovr} , B_{ovr} , A_{ovr} , P] , A, B, M } and its *relocation context:* (W,X,Y,Z), together with the actual segments themselves.

Note that the three overflow indicators (M_{ovr} < B_{0vr}, and A_{ovr}) are combined with the thirteen bits of the P register to form the first word of the working context. See Section 2.2 for more on overflow indicators.

Instructions are available (eg functions 29 and 30, as described in the Appendix) which exchange the current working context with one held in store. This provides powerful sub-programming facilities.

3.2.8. Communication with Peripherals

A peripheral device may only be accessed when fetching the operand for a function, though there is no restriction on which function is used. Thus arithmetic may be performed directly with such an operand, without the need to store it first or to obey any other instruction. Both store modules and peripheral devices may accept or reject requests for access. In the case of store rejection, this is considered an error and, as such, causes an error interrupt. It is to be expected, however, that peripheral devices will sometimes reject a request. To allow for this possibility, the following procedure is adopted. If the request is accepted, then the function is performed normally but the next instruction in sequence is skipped. If the request is rejected, then the function is not performed. Instead, a rejection qualifier is loaded into the B register and the next instruction in sequence is executed.

3.3. Interrupts, Special State and Normal State.

3.3.1. Scene-setting.

The Processor can operate in one of two states, *normal* or *special*. In general, all user programs will operate in normal state. The processor enters *special state* only as a result of an interrupt, which may originate externally or internally. *Special state* code has priviledged access to all storage locations and all peripherals. Another difference between special and normal state is that whereas normal state is re-entrant, special state is not. This allows special state code to be entered and left efficiently, so that interrupts can be serviced at high speed. Special state code is restricted to the first 8K of store.

Each *special state* routine is associated with one or more sets of four dedicated store locations which hold its working context while it is inactive. In a special state working context M_{ovr} is replaced by an interrupt reject indicator I_{rej}.

On the occurrence of an interrupt, the working context of the current *normal* state program is exchanged with that stored in the locations dedicated to that interrupt. This operation is performed entirely by the hardware. Special state routines have no associated relocation context and cannot themselves be interrupted. They also have privileged access to all peripheral devices and storage, and to the relocation registers.

3.3.2. External Interrupts.

A peripheral device can interrupt the processor which automatically performs an exchange of data words with that device, to trigger off a *special state* routine. *Special state* code will cause interrupts from all peripheral devices to be rejected, by setting their *global inhibition*, and further occurrences of one type to be rejected, by setting an *interrupt specific reject indicator*.

3.3.3. Hesitations.

The processor acts as a store multiplexer to allow any of its peripheral devices to use its facilities for accessing all of the direct storage available to the processor itself. This is the only method by which peripherals can access internal semiconductor store. *Special state* code can cause hesitations from all peripheral devices to be rejected, by setting their *global inhibition*.

The effect of this depends on the characteristics of the addressed channel, which may respond in one of the following ways:

1. Inhibit interrupts and hesitations,
2. Reject interrupts and allow hesitations,
3. Inhibit interrupts and allow hesitations.

3.3.4. Software Interrupts

A normal state program may call upon special state service routines by generating one of 64 software interrupts available in the processor.

3.3.5. Error Conditions

The following error conditions are trapped, forcing entry to appropriate routines dependent upon whether a *normal state* or a *special state* program was running at the time of the error:
store rejection
invalid instruction.

3.4. Instruction set.

3.4.1. Overall listing.

The 32 operations are listed below. Functions 24 to 30 involve complex operations that are best explained in detail in the Appendix.

Function (op code)	***Description***	***Mnemonic***	***Effect***
0	ADD A	ADA	A := A + Q
1	SUB A	SBA	A := A – Q
2	LOAD A	LDA	A := Q
3	STORE A	STA	Q := A
4	ADD B	ADB	B := B + Q
5	SUB B	SBB	B := B – Q
6	LOAD B	LDB	B := Q
7	STORE B	STB	Q := B
8	ADD M	ADM	M := M + Q
9	SUB M	SBM	M := M – Q
10	LOAD M	LDM	M := Q
11	STORE M	STM	Q := M
12	ADD P	ADP	P := P + Q
13	SUB P	SBP	P := P – Q
14	LOAD P	LDP	P := Q
15	STORE P	STP	Q := P
16	ADD W	ADW	W := W + Q
17	SUB W	SBW	W := W – Q
18	LOAD W	LDW	W := Q
19	STORE W	STW	Q := W
20	MULTIPLY single	MLS	A := A x Q
21	MULTIPLY double	MLD	[B,A] := A x Q + B
22	DIVIDE	DIV	A := [B,A] / B
23	EXCHANGE	EXC	A := Q; Q := A
24	SHIFT	SFT	See Appendix 3.1.1
25	COPY	CPY	See Appendix 3.1.2
26	TEST	TST	See Appendix 3.1.3
27	INTERFACE CONTROL	STS	See Appendix 3.1.4
28	SUBROUTINE ENTRY	SRE	See Appendix 3.1.5
29	PROGRAM LINKAGE	ENT	See Appendix 3.1.6
30	EXECUTIVE CALL	EXT	See Appendix 3.1.7
31	INVALID	-	(violation)

3.4.2. Instruction times.

Initially, Modular One processors were fitted with a fast core store (0.75 microsecond cycle time). From about 1970, semiconductor memory was installed. The type 1.14 Modular One Processor is normally used either with a type 126 semiconductor store module or a type 127 semiconductor store module. At the time of writing (January 2023) the cycle times of these memory systems has not come to light.

Quoting from Reference 3.1, typical instruction times for sample instructions, in microseconds, when using addressing mode 2 (direct) are:

Instruction	*Direct addressing, type 126 store*	*Direct addressing, type 127 store*
ADD	1.8	2.4
MLS	3.6	4.2
MLD	3.3	3.9
DIV	5.1	5.7
Shift single-length left 1 place	1.9	2.6
Tally logical right 15 places	6.3	6.9
Function 25 logical	1.9	2.6
ENT	6.5	8.1

Appendix 3: further details of the complex instructions.

A3.1.1 Function 24, SHIFT.

Bit	*Bit-value*	*Operand value*	*Significance*
0 – 3	n	n	Specifies the maximum number of places to be shifted, as a binary number, n, such that 0 < n < 15
4	0 1	0 16	Shift right Shift left
5	0 1	0 32	Arithmetic logical
6 & 7	0 1 2 3	0 64 128 192	Single length Double length Justify Tally
8 - 15	Any	-	(These bits are not used)

The operand-value column, n, gives the contribution of the field towards the total value, Q, of the operand. For example, a justify logical right shift of 6 places is specified by an operand of:
0 = 128 + 32 + 0 + 6 = 166.

Explanation of Tally.
The Tally operation can be either a shift left or right, arithmetic or logical, depending upon the values of operand bits 4 and 5. A *Tally shift* counts in register B, the number of times the least-significant digit of A changes from 0 to a 1, or vice versa, during n shifts. Prior to the operation, register B is cleared. Then, at each subsequent stage of the shift, register B is incremented by 1 if the new value of the ls digit of A is going to differ from the old.

As an example, consider A = 1100 1110 1011 1000.
If a tally arithmetic right shift of 13 places is specified, A is shifted to
A = 1111 1111 1111 1110 and the B register is set to
B = 0000 0000 0000 0110.

A3.1.2. Function 25, Inter-register operations (COPY).

This function provides comprehensive facilities for inter-register arithmetic and logical operations. The detailed actions performed are specified by individual fields within the operand, which are arranged so that the most commonly occurring combinations can be accommodated by a literal.

The function's operand defines :

(a) the source registers that will supply two operands, denoted in the Table below by S and T;

(b) one of eight operations to be performed on S and T to yield a result, denoted by R in the Table, and the result registers that will hold R.

If no registers are designated to supply S (or T), that operand is taken to be zero, while if more than one register is designated to supply S (or T), that operand is the logical OR of the contents of these registers. Several registers may be specified as result registers and, in this case, the result is placed in them all.

The A, B and M overflow indicators are not affected by this function, even if the A, B or M registers are involved in arithmetic operations which generate results outside the arithmetic range.

The following table shows the operand significance for function 25. Note that the symbol | signifies *logical OR* and the pair ≡/ signifies *Exclusive Or* (ie *Not Equivalent To*).

Bit	***Bit value***	***Operand value***	***Significance***
0 & 1	0 1 2 3	0 1 2 3	- S := S | A S := S | M S := S | W
2	0 1	0 4	- S := S | B
3	0 1	0 8	- T := T | B
4	0 1	0 16	- A := R
5	0 1	0 32	- B := R
6 to 8	0 1 2 3 4 5 6 7	0 64 128 192 256 320 384 448	R := T & S R := T & (~S) T := T | B R := T ≡/ S R := T + S R := T – S – 1 R := T – S R := T + S + 1
9	0 1	0 512	- S := S | P
10	0	0	-

	1	1024	T := T | P
11 & 12	0 1 2 3	0 2048 4096 6144	- T := T | A T := T | M T := T | W
13	0 1	0 8192	- P := R
14	0 1	0 16384	- M := R
15	0 1	0 32768	0 W := R

All fields are mutually independent and this allows the use of more than one register as both source and destination.

Example 1:
If the initial operand is the literal 57, ie the binary pattern 00111001, then the effect is:
S := S | A, which is equivalent to S := A
T := T | B, which is equivalent to T := B
R := T & S, which is equivalent to R := A & B
A := R, which is equivalent to A := A & B
B := R, which is equivalent to B := A & B

Example 2:
If the initial operand is 18568, ie binary pattern 0100100010001000, then the effect (assuming S and T are initially clear) is:

T := T | B then T := T | B R := T ≡/ S then T := T | A then M := R

T := T | BA | B
R := (A|B)=K>
M := R

The effect is to place A NOR B in register M.
The same effect could be achieved by the operand 16709,
i.e.
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 1
S : = A | B (4+1)
T := 0 (0)
R : = 0 - (A | B) - 1 (320)
M := R (16384)
N.B. -(AIB) = ~(A|B) + 1 so that —(A|B) - 1 = ~(A|B)

(For more information, see section 4.25, page 115 onwards, in Reference 3.1).

A3.1.3. Function 26, Conditional Skip (TEST).

This function provides comprehensive facilities for testing the arithmetic status of the A, B and M registers, followed by a skip of some instructions if the test fails. The detailed operation is specified by the individual fields within the function's operand, which are

arranged so that the most commonly occurring combinations can be accommodated by a literal. The operand defines:

a set of arithmetic conditions to be tested;
a register to be incremented after the test;
a number of instructions to be skipped if none of the selected conditions is satisfied before incrementation.

The fields specifying the tests to be performed are interpreted independently and any combination of tests may be used. The skip will be performed only if none of the tests is satisfied before any register-incrementation. If no tests are specified, none can be satisfied; therefore, in this case, the skip is always made. Any incrementation is performed regardless of whether the skip is made or not; but no overflow indicator is set if the result of this addition lies outside the arithmetic range.

This function may be used to test the A, B and M overflow indicators. If any of those specified are set, then the condition is regarded as being satisfied and no skip is made. Any indicator which is tested by this function is reset, but others will be left in their current condition. This is the only method of resetting the overflow indicators without performing a sub-program entry or exit. For *special state* programs, the M overflow indicator is replaced by the interrupt rejection indicator.

The following table shows the operand significance for function 26.

Bits	***Bit-value***	***Operand value***	***Significance***
0 & 1	0	0	-
	1	1	A := A + 1
	2	2	B := B + 1
	3	3	M := M + 1
2	0	0	-
	1	4	A < 0 ?
3	0	0	-
	1	8	A = 0 ?
4	0	0	-
	1	16	A > 0 ?
5	0	0	-
	1	32	B = 0 ?
6	0	0	-
	1	64	M = 0 ?
7	0	0	-
	1	128	M > 0 ?
8	0	0	-

	1	256	M < 0 ?
9	0	0	-
	1	512	B < 0 ?
10	0	0	-
	1	1024	B > 0 ?
11 & 12	0	0	P := P + 1
	1	2048	P := P + 2
	2	4096	P := P + 4
	3	6144	P := P + 8
13	0	0	-
	1	8192	A_{ovr} = 1 ? A_{ovr} := 0
14	0	0	-
	1	16384	B_{ovr} = 1 ? B_{ovr} := 0
15	0	0	-
	1	32768	M_{ovr} = 1 ? M_{ovr} := 0

Note that the incrementing of P is only dione if none of the tests is satisfied. All fields in the operand are mutually independent. For example A = 0 and B = 0 can be tested together. The skip in this case will not be made unless A and B are both non zero; if either is zero then the instruction following the test is obeyed as normal. Note also that the three conditions ≠ 0, ≥ 0 or ≤ 0 may be tested by invoking two test indicators. Thus, the test A ≠ 0 is accomplished by setting bits in the operand that specify both A < 0 and A > 0 within the same instruction.

The following example illustrates the use of the operand with the *Function 26* instruction. If the operand is the literal 59, ie 00111011, then the action is:
skip 1 instruction unless B = 0 or A > 0 and increment M by 1.

A3.1.4. Function 27. Interface control (used for inhibiting interrupts and hesitations).

The operand for Function 27 specifies the processor's response to interrupts and to hesitations, and provides a means of setting the Interrupt Reject indicator. This function is only valid while the processor is in *special stat*e. If it is issued while in *normal* state, it results in an internal interrupt occurring to the context stored in locations 2,4 onwards, indicating an invalid function.

The operand for function is treated as a literal, with the literal's bits being interpreted as follows (see next page).

Bit	*Value*	*Operand*	*Meaning*
0 and 1	0 1 2 3	0 1 2 3	- - Set *inhibit normal state interrupts* status Set *allow normal state interrupts* status
2 and 3	0 1 2 3	0 4 8 12	- Set *inhibit hesitations* status until *specia! state* is left. Set *inhibit hesitations* status Set *allow hesitations* status
4 and 5	Any		(Not used)
6	0 1	0 64	- Set processor non-operational.
7	0 1	0 128	- Set interrupt-specific reject indicator I_{rej} := 1.
8 to 15	Any		Not used. (Since these bits are invariably zero, it is always possible to specify the operand as a literal.

Notes.
(a). The processor will always inhibit interrupts while in special state. If the processor is set to inhibit interrupts, any incoming demand that is found to be an interrupt will be inhibited with a special busy signal to the channel, which may in turn generate a rejection qualifier.
(b). If the processor is set to inhibit hesitations, any incoming demand that is found to be a hesitation will be rejected with qualifier 4. If the processor is set to allow hesitations, any incoming demand that is found to be a hesitation will be dealt with normally, regardless of whether or not the processor is in normal state.
(c). The processor may set itself non-operational by executing a function 27 with Q6 = 1. if the processor power supplies are turned off, or fail, then after a given period of time the processor will be set non-operational, even if no function 27, with Q 6 = 1, has been executed.

A3.1.5. Function 28: subroutine entry.
This function provides a conventional sub-routine entry facility. The more powerful sub-program facilities of the Modular One are provided by the program linkage instructions described in Appendix A1.6 and A1.7 below.

Function 28 places the current value of the Program Counter P into the B register and sets the new value of P to be a value that depends upon the bits in the operand Q - as shown in the Table below.

Value of operand Q	*Resulting action*
$2^{13} > Q \geq 0$	B : = P P : = Q
$Q \geq 2^{13}$	B := P P : = Q - m * 2^{13} m is a positive non-zero integer such that $2^{13} > Q - m * 2^{13} \geq 0$
Q < 0	B : = P P := Q + m * 2^{13} m is a positive non-zero integer such that $2^{13} > Q + m * 2^{13} \geq 0$

A3.1.6. Function 29: program linkage type 1 (ENT).

This function operates differently in each of the two states:

normal state: switch the working context & enter code at point specified by the operand, Q;

special state: switch the relocation context.

The actions relate to the Modular One concepts of Segments and Context, as desribed in sections 2.6 and 2.7 above.

Function 29 in normal state: sub-program entry.

In summary, the contents of the A , B, M and Overflow registers are exchanged with the contents of words 1, 2, 3 and 4, respectively, of the local workspace (the Y segment) and the new value of the program counter P is set to a value determined by the value of the Function 29's operand.

The new value of P is as defined in the following Table.

Value of operand, Q	***New value of P***
2^{13} 1 > Q ≥ 0	P := Q
Q ≥ 2^{13}	P := Q - m * 2^{13} m is a positive non-zero integer such that 2^{13} > Q - m * 2^{13} ≥ 0
Q < 0	P := Q + m * 2^{13} m is a positive non-zero integer such that 2^{13} > Q + m * 2^{13} ≥ 0

When the contents of the A , B, M and Overflow registers are exchanged with the contents of words 1, 2, 3 and 4, respectively, of the local workspace (the Y segment), a check is made to ensure that the four addresses do not point outside the Y segment bounds, or to a peripheral. If one does, an error interrupt made to the context stored in locations 2,4 to 2,7 indicating an address violation.

Function 29 in special state: switch relocation contexts.

This function exchanges the contents of the W, X, Y and Z registers (the relocation context of the current program) with the contents of four consecutive locations of store, as defined by the function's operand, Q, whien considered as a 16-bit positive address. In the following explanation, [Q] denotes the initial contents of the store location pointed to by the Function 29's operand. A prime ' is used to denote the new contents of a store location, or of a register, after the Function 29 has been obeyed. The four consecutive store locations are denoted by Q, Q+1, Q+2 and Q+3.

[Q]' := W; W' := [Q]
[Q+1]' := X; X' := [Q+1]
[Q+2]' := Y Y' := [Q+2]
[Q+3]' := Z Z' := [Q+3]

Function 29 is the only method of changing the settings of the X, Y and Z registers.

A3.1.7. Function 30: program linkage type 2 (EXECUTIVE CALL).

This function operates differently in each of the two states.

In normal state:
The effect is to exchange the contents of the P, A, B and M registers with the contents of the first four locations of local workspace. In particular:

If Q = 0, switch the working context and enter code as defined by the new P register value. This is equivalent ot a sub-program link within normal code.

If 1 ≤ Q ≤ 255, switch the working context and enter *special state* code as defined by the new P register value. This is equivalent to an Executive call.

If Q is outside the range 0 -> 255 then cause an *invalid instruction interrupt* to the context stored in locations 2,4 to 2,7.

In special state:
The effect is to exchange the contents of the P, A , B and M registers with the contents of four consecutive locations of memory, as defined by the operand, Q. Normal state is then entered. In particular:

switch the working context and enter normal state code as defined by the new P register value.

References for section 3.

3.1. *Modular One functional specification: the 1.14 Processor*. January 1974.

3.2. *NAL fact sheet*. CTL technical publication 382/17/4C. Issue 2, November 1976.

3.3. *Programmers' Handbook – draft 1.11: the Address Modes.* P J Clark. Document 70/707/G, October 1970.

3.4. *Modular One.* 25-page glossy brochure with art-work. Publication date not given but deduce it was 1971 or 1972.

The above are published by Computer Technology Limited, Eaton Road, Hemel Hempstead, Hertfordshire, England.

4. Software.

4.1. The Executive and Operating Systems.

The smallest Modular One systems are designed to operate with an Executive program, which contains facilities for the control of peripheral devices, for organising and allocating storage and processing time, for inter-program communications, for switching from one program to another and for dealing with the power on/off interrupts and error conditions. The Executive also contains service routines, accessible by software interrupts, which enhance the hardware facilities.

The X, Y and Z memory segmentation registers, together with the two execution states (*Normal State* and the non-interruptible privileged *Special State*) mean that the Executive can act as a self-protecting Operating System kernel.

As for full-blown Operating Systems, CTL provided MODUS 4 which gave “efficient multi-programming, real-time response, program protection and many facilities including Virtual Memory capability previously only associaterd with large processor systems” [Reference 4.1]. MODUS 4 was modular and highly configurable, able to be tailored to suit the needs of individual customers.

More details to come.

4.2. Programming languages and compilers.

At the lowest level, an Assembler called NAL is provided for the Modular One. There are the usual range of text editors and debugging tools.

High-level compiles include CORAL, FORTRAN IV and BASIC. CTL provides a range of area-specific software applications packages.

More details to come.

Reference for section 4.

4.1. *Modular One.* 25-page glossy brochure with art-work. Publication date not given but deduce it was 1971 or 1972. Published by Computer Technology Limited, Eaton Road, Hemel Hempstead, Hertfordshire, England.