3- ICT/ICL 1900 Range Software

During the period of development of the1900 Range, the Software activity evolved from a Sales Support Aid into a major computer industry driver, which, towards the end of the period, was absorbing development resources in excess of the hardware.

This Section will outline its evolution in the context of the ICT/ICL 1900 Range

3.1 – The FP 6000 Software (Before the 1900 Range)

The following is the relevant extract from the "FP6000- Report on Visit to Ferranti-Packard" in April 1963, detailing present or planned software while recommending the adoption of the FP6000 as the Ferranti 1900 (To be noted that Executive was considered an item of software in Ferranti-Packard, while it became part of each "processor" development in the ICT 1900 Range and, together with Test Programs and Diagnostics, was no longer listed by ICT in the general software lists):

*"7. **Programming***

*7.1 **General Software***
The software which either exists or to which Ferranti-Packard are committed is as follows;-
(1) Executive
This is the F-P name for what we would call the supervisor. It is written in modular form and provision will be made for controlling up to 20 peripheral equipments. In addition it deals with extracodes and timesharing activities. For any particular system only those parts of the executive which are required will be supplied. The version for the paper tape system is virtually ready. The dead-line is 25th April when it is needed for the N.R.E. machine (the P.R.B machine is a special case and does not require executive). This version will occupy 960 words.
The typewriter is used for communication between executive and the operator and a set of stereotyped messages exists (e.g. SUSPEND, GO, ALTER, REVISE PRIORITY).
(2) Debug
This is a program that may be regarded as an extension to executive. It assits in the development of programs and can be 'steered' to permit, for example,
'Single-shot' working with print out,
Monitoring at prescribed points in a program, or
Monitoring at successful jump instructions
This program has been flow-charted and partially written and is required by the end of April by the N.R.E.
(3) Assembler
This is a simple input scheme for machine language programs. It was referred to as 'a poor man's Pegasus, Initial Orders with some extra features'. It is a 'load and go' assembler with a 1-1 transformation from the written form of instructions to the machine representation. Names(of up to

k characters) are allowed for symbolic addresses and provided that all symbols are pre-defined, simple arithmetic can be done on them.
The assembler enables library routines to be incorporated into a program by means of the directive "Library" followed by the names of the routines required. Other directives specify the amount of storage required by the program, the peripherals required and can allocate the priorities between the main program and up to 2 sub-programs which can time- share with the main program.
This assembler appears to provide an adequate input scheme for machine language programs. It has been checked out and is used at present as the standard way of taking in programs that are being developed.

(4) Fortran Compiler

A Fortran compiler, modelled on the 1620 Fortran but being made compatible with Atlas Fortran, is being written for the N.R.E. It is scheduled for completion by October, 1963. This date may be a little optimistic but the commitment to the N.R.E. should ensure that it is completed.

(5) FP6000 Autocoder

This is a more sophisticated input than the assembler. It handles mnemonic forms for the functions in machine language and has no restrictions on symbolic addresses. It also enables macro-instructions to be assembled. Further features which are planned include the ability to translate simple logical and arithmetic statements.
No precise specification of this language exists and its status will be affected by any decisions regarding the implementation of COBOL. It would appear that F-P have in mind a language which should be relatively easy to implement and which would make a useful alternative to a much more expensive COBOL or NEBULA compiler.
No firm dates were mentioned for completion of this project.

(6) Library

At present the library consists of routines concerned with the input and output of numbers, fixed to floating conversions, floating point operations, the usual mathematical functions and certain double-length routines. In all some 40 routines have been written and are being tested. The library list is appended.

(7) Diagnostics

This is the F-P word for engineers test programs. These have been written for the paper tape system and cover
Core store tests
Individual function tests
Paper tape reader tests
Paper tape punch tests
Typewriter input and output tests
Further tests will be written to cover other parts of the system as later orders make them necessary.
No attempts have been made to write diagnostic tests in the sense that they are used at West Gorton.

(8) Matrix Interpretive Scheme

It is intended to write a matrix interpretive scheme based on the Deuce scheme rather than the Pegasus one. The main difference is that storage can be freed when no longer required by the scheme.
Little work has been done on this scheme to date.

(9) ALGOL compiler
It is believed that an ALGOL compiler will be written in collaboration with the Saskatchewan Power Corporation. It is expected that the Corporation will supply a programmer who has previously written an ALGOL compiler for another machine to work in Toronto with F-P for one year."

3.2 - ICT 1900 Range Software – The Start

The following is an extract from the talk that Peter Hunt gave at the London 1900 Seminar in May 1996:

"In late autumn 1964 I was asked by Arthur Humphreys to take charge of the production of the software for the 1900 computer range. On reflection, I am not sure the word "software" was used: nowadays that's what it would be called.

At the time I was Head of the Bracknell Laboratories running several interesting hardware and software projects, with large penalty clauses. I had about 12 years programming experience, and had learnt the hard way that software projects were easy to conceive but difficult to implement, were invariably late on delivery and usually exceeded budget.

I asked what software had been promised for 1900, and was given a copy of a document — I think it was document number 1024 — which listed what would be available and by when on the hardware of the 1900 as it was envisaged at that time. I have never found out who wrote the document. It was a phenomenal list of software on various configurations including all types of different peripheral equipment.

After reflection, I decided that I would only accept the appointment under certain conditions:

- *I would be allowed to revise document 1024 with more appropriate delivery dates;*
- *I would be given the appropriate resources to do the job, including adequate competent experienced programmers, and adequate computer hardware facilities under my own control on which to develop the software; and*
- *that I report directly to Arthur Humphreys (so that I could be sure that if I had problems they would hopefully be sorted out quickly without going through layers of management!).*

Arthur accepted these conditions, and then the real work began.
First, I rewrote document 1024 and reissued it with more realistic delivery dates. This was not difficult in itself, but it did involve re-educating the sales force, which had been working with the previous edition. I had to spend time in meetings with senior members of the sales force, and give lectures to the junior members, assuring them that the new dates would be kept, and that there would be no further slippages during the lifetime of the project.
Second, we started work immediately on recruiting 100 programmers.

Third, it was arranged that the necessary hardware would be delivered to "Programming Division" (as we were then known). This included every model of processor and at least one of every type of peripheral that was likely to be delivered to customers.

I organised the production effort into a number of divisions responsible for specific software areas:
- *Operating Systems*
- *Compilers (scientific and commercial)*
- *General Purpose Software (eg Plan assembler language, housekeeping software)*
- *Applications Programs*
- *Services (including supply of computer time, quality assurance and issue of software)*

So there were five divisional managers. I met them every Monday morning to sort out problems and to consider progress on all matters.
At the time, software engineering was either non-existent or in its infancy. The divisional managers decided how each of their projects should be organised, how they would be implemented (there was no BS standard in use) and how they would monitor progress against the promised dates (Pert was in fact one of the applications programs we were implementing).

One of the aspects of the project that worried me most was the problems that might arise when we started to issue all this software to customers all over the world. The originating programmers would have tested it to the best of their ability but, as we all knew, the software would still contain bugs when issued. We wanted these reduced to a minimum.

We therefore set up a primitive quality assurance group. Its task was to take software from the production divisions once they said it was ready for release and, using only the documentation available to customers (we had a separate group, not reporting to me, of technical authors producing manuals and user guides), to use the package as fully as possible, imitating as far as they could the day-to-day usage by customers.
When they discovered errors, the package was referred back to the originating division for correction. The quality assurance team then re-tested the corrected package, and this iterative process continued until the team was satisfied with its correctness. Only then was the software released.

It was not long before 1900s were being delivered in quantity all over the world, and the software distribution system had then to be organised properly.
We decided to issue the software on magnetic tape as the general rule. This meant we had to have fallback arrangements for those installations which did not have magnetic tape transports, such as the 1900s which used cassette tape.

Once issued each software package had to be supported, as we would now put it. When a customer reported an alleged error, this call (or more likely a letter) would be dealt with initially by a front line support force in another division (not reporting to me). This was because in many cases the customer simply needed assistance in the use of the package.
When the support team thought the customer had discovered a genuine error they referred it to us. We investigated and reported back. If it turned out to be an error on our part a "software notice" would be issued to all customers with the following information:

1. description of error;
2. when it would be corrected (release number and date due);
3. what to do in the meantime.
This all seems very standard today, with our modern help lines, but 30 years ago we were breaking completely new ground."

3.3 – Programming Aids (Software) in 1966

The following is an extract from an ICT brochure dated November 1966, listing the "Software" of the 1900 Range:

"Programming aids

No program for the 1900 Series will be written in machine code. The extensive range of central processors and peripheral devices has been matched by an equally extensive range of programming aids. This range includes:

Plan- An assembly system specially prepared for the 1900 Series employing mnemonics.

Cobol- An international programming language of considerable power and flexibility for general commercial applications.

Rapidwrite- The I.C.T. simplification of COBOL.

EMA- The Extended Mercury Autocode for the solution of mathematical and scientific problems. It is simple to use, and widely employed in the scientific world.

Fortran and Algol- International autocodes for mathematical and scientific tasks.

NICOL- A simple commercial programming language developed by I.C.T., it simplifies the task of transferring jobs from a manual or automatic accounting system to a 1901 installation. Its straightforward format and few operational commands enable users to translate their requirements into computer terms after as little as four days instruction. Although developed for use with the 1901, series compatibility ensures that programs written in NICOL can also be run on any other 1900 Series computer system.

Sub-routines- A comprehensive library of commercial and mathematical sub-routines is available.

Operating Systems- To optimise the running of jobs, I.C.T. has devised a series of sophisticated operating systems to schedule the work, to organize and control the files and to control communications with the users.

Packaged Programs- Complete packaged programs have been devised by I.C.T. which require only the insertion of users' parameters. These packaged programs cover such tasks as: Linear Programming, Matrix Algebra, Survey Analysis, Systems Control, PERT, Inventory Management, Transportation, Regression Analysis. **"**

3.4- The Operating Systems (George).

Key software products of the 1900 Range were the George Operating Systems

3.4.1- The Start

The following is an extract from the talk that George Felton gave at the London 1900 Range Seminar, describing the background and origins of the George Systems:

"Preliminaries

The successful George systems were based on original ideas developed between 1959 and 1964 for the Ferranti Orion computer. I shall have to outline these to start with.

Orion Monitor Program (OMP)

Thinking about the Orion system started late in 1958. From Jan 1959 intensive work in Ferranti on new computers led to Atlas and Orion. Development of the operating system (OMP) took place in parallel with that of the hardware (which was heavily oriented to users, ie programmers and operators). In March 1959; a presentation of both machines was made to Harwell. At this time much effort was still going into Pegasus and Mercury.

We designed simple but effective features in the Orion hardware to allow simultaneous running of independently written programs – we called this Time-Sharing but it would nowadays be called pre-emptive multi-tasking, with limited multi-threading. There were privileged instructions to allow efficient time sharing or multi-programming. There were hardware datum and limit registers to prevent any program from accessing any part of the core outside its own entitlement – this prevented accidental interference between jobs and was considered vital for sanitary time sharing (paging was introduced later and provided a different but equally effective way to prevent interference between jobs). There were also arrangements for a program to access only its own peripherals, referring to them by its own private set of addresses. Each program had its own set of registers (accumulators, modifier registers, etc). All these features seemed obvious at the time and they were cheap to include in the hardware. OMP didn't include any multi-access features – the idea was invented later at MIT as Project MAC.

On 12 Nov 1959 there was a press conference on Orion in West Gorton. On 14 Jan 1960 I gave a Cambridge colloquium on the architecture, later repeated in other universities (London, Glasgow, Newcastle, Edinburgh, and Oxford). And in May 1960 I described Orion at the Australian Computer Conference in Sydney. (Orion 2 design was started in 1961.) In Feb 1961 simulation was used to help design the time-sharing system – see the paper by H P Goodman in the September 1961 issue of the Computer Bulletin.

In June 1962 we started shift work developing OMP and Nebula on the Orion prototype in West Gorton. We started running time-shared programs on the prototype Orion by May 1963, when the first (Turitz) Orion started acceptance tests.

The George Development

In about Dec 64 (?) Peter Hunt was made responsible for software development in ICT and a request was received from Chris Wilson (responsible for selling the 1906/7) for an operating system on the lines of OMP for the 1906/7. On 1 Jan 65 I called a meeting with Henry Goodman, John Fotheringham, and Henry Goldberg to discuss this requirement. There was intensive activity on specifying the system, learning about the 1900 hardware, moving people, recruiting people, running courses, reorganising, taking on consultants, etc. The 1906/7 was the main focus for operating systems. Work on developing OMP Mark 2 continued through all this.

3.4.2- George Operating Systems for the ICL 1900 Series Computer Range.

(Summary Description by H. P. Goodman January 2004).

1 Historical Introduction

In December 1964 the company was reorganised so that all software development was moved into Programming Division managed by P.M. Hunt. A branch was set up (Operating Systems Branch) which would eventually design and build a major operating system for the 1906 and 1907, these being the top end of the 1900 range as originally announced. Initially the branch was staffed from those who were engaged in Orion built in software. The Orion work was nearly complete and the intention was that these staff would build the 1906/7 Operating system as they became free.

The initial outline specification of the system envisaged a batch processing system incorporating the time sharing features of Orion and the spooling features of Atlas; the core of the system would be a file store with a minimum requirement of a 512k word drum. After a few months the system was named George as a compliment to the Department manager, George E. Felton. The official explanations were that it stood for General Organisational Environment and that it was analogous to the autopilot on Second World War era aircraft, popularly known as George.

In July 1965 a major seminar was held at N.P.L. describing Project MAC at M.I.T . This was a system which had been developed on a highly unsuitable IBM computer to provide a primitive multi-access system "a 7090 in your office" which allowed up to 32 simultaneous users on teletypes to develop and run programs. This was obviously going to be a major way of using computers and ICL immediately decided that it needed a product in this area for the 1906/7. This was to be called MOP (Multiple Online Processing) and the project was added to the workload of the team developing George. In November 1965 I attended the Fall Joint Computer Conference in Las Vegas where the major topic was Multics, the new multi-access system being designed at M.I.T. I brought back a number of papers on Multics and many of its ideas were incorporated into George, particularly the tree-structured file store. Around this time it became clear that rather than produce two separate systems, George and MOP, it was preferable to build a combined system which provided both batch and multi-access facilities; MOP thus ceased to exist as a separate system but the term continued to be used to describe the parts of George dedicated to multi-access facilities.

Towards the end of 1965 it became clear that there was a large gap between the simple time-sharing facilities provided by Executive and the sophisticated facilities to be provided by George. Marketing therefore requested two simpler and earlier operating systems than George. It was decided to rename George as George 3 so that the simpler systems could be called George 1 and George 2 respectively. George 1 would run a single stream of programs with job control facilities to replace operator action. George 2 would add spooling facilities to George 1.

2 Basic Features of George 3

The main components of George 3 were:
(a) The command language which was used both for batch job descriptions and by on-line users at MOP terminals,
(b) The file store which is the heart of the system containing files of many types, including those read in from basic peripherals.
(c) The various scheduling subsystems including facilities for swapping and off-lining.
(d) MOP and other remote access facilities
(e) Accounting and budgets

Unlike George 1 and 2 which were built on top of a standard Executive, George 3 effectively controlled the whole machine. There still had to be an Executive whose main functions were low level control of various peripherals and concealing hardware differences between various members of the 1900 range. This Executive had no user-visible functions. The George team, in Putney (London), defined the interface between George and Executive. the executive was implemented by programmers close to the hardware manufacturers in West Gorton (Manchester) and Stevenage.

George 3 was written in an Assembly Language. It turned out that the standard 1900 Assembler (PLAN) could not be used since facilities to use extended mode programming, hardware only available on 1906/7 initially, would not be available in time for development of George 3. A cut down assembly language called GIN (George Input) was developed and this evolved into something particularly suitable for George 3 with the inclusion of sophisticated macro and conditional assembly features.

3 The Command Language

The George 3 command language consisted of a number of built-in commands of various types which was augmented by a powerful macro facility. Commands had a variable number of parameters, normally separated by commas, and there was considerable flexibility as to whether parameters appeared in fixed positions or were specified by keywords, in which case they could appear in any order. The macro facility could be nested to any number of levels and recursive macros were allowed and often used. The fact that the same command language could be used for both batch jobs and on line work with MOP proved to be very important; particularly for testing programs on-line which could then be run in batch mode with no change to the job description. It is interesting to note that this facility was not available on IBM machines until 1974. Many users built up complex libraries of macros which could call each other. In some cases this became almost a programming language which could be used to built up complex jobs from a sequence of simpler programs.

Each built in command had a 2 letter abbreviation. Over the years, as the number of commands increased, it became more and more difficult to find an appropriate name for a new command which yielded a plausible 2 letter abbreviation which had not already been used. Similar problems arose with command syntax as it got extended many times to meet new requirements. The development of later versions of George 3 was largely customer driven; the development team published a list of suggested enhancements periodically and a committee of users assigned priorities to these which thus determined what we would be added to the next mark of the system.

4 The File Store

The original specification was for George 3 to work with a minimum backing store of 512k words, this being the size of a large drum available at the time of the original announcement. As the system developed this proved to be unrealistic as more and more uses were found for direct access devices and as larger and larger exchangeable and fixed disc systems became available.

The file store provided spooling (aka off-lining) facilities for all basic peripherals (e.g. paper tape and card readers and punches and line printers) so that input documents were read into the file store and then passed to the object program one line at a time; similar output for punches and printers went to the file store and were output later by George.

There was an "upwards compatibility" commitment which meant that George 3 had to be able to run any program written to run under Executives on smaller machines of the 1900 range. Such programs often addressed peripherals directly using, for example, slightly different PERI instructions to output to line printers, card punches and paper tape punches. In order to support such programs files in the file store had to have a file-type such as line printer or paper tape reader and, although the file was actually on backing store, it was read (or written to) using appropriate PERI instructions. George also allowed some flexibility here, e.g. allowing a file output to a line printer to be read by a program expecting card input. Programs written specifically to run under George 3 would refer to disc files rather than basic peripheral files.

The file store was hierarchical with a tree structure of directories of arbitrary depth. Each directory had an owner who also owned all the files in the directory. A subdirectory could either belong by default to the owner of the superior directory or could itself be assigned to a different owner. Each user had a home directory and file names could be given relative to the current directory, the current user's home directory or absolute, from the root directory. There was an elaborate system of user traps to allow users to read or write to files belonging to other users. There was an incremental dumper program called in at regular intervals which dumped copies of files which had been created or modified since the last dumper run to a dedicate magnetic tape known as a dump tape. This allowed restoration of individual files or the whole file store if this turned out to be necessary after a system crash.

There was also provision to handle disc files that were not part of the file store, known as exofiles; these were needed by programs which wished to handle the physical properties of large files, e.g. to minimise head movement. Thus large main files, especially indexed sequential and random files, were usually exofiles. File store files were mainly used to handle the mass of small files where the user did not need to know how they were organised.

5 Scheduling

The scheduling mechanisms within George 3 were on three levels:
(a) The High Level Scheduler
(b) The Low Level Scheduler
(c) Executive Time sharer.

The function of the High Level Scheduler was to examine the queue of jobs waiting to be run at any one time and select which of those need to be made active; i.e. known to the Low Level Scheduler. The object was on the one hand to generate an efficient mix of jobs to optimise the use of the system's facilities and on the other hand to process jobs in accordance with the system manager's priorities. There were a number of installation parameters which could be set to define the installation's policy; also jobs were given different urgencies which the High Level Scheduler would take into account. The main part of the High Level Scheduler was written as a subject program; this was an object program with a slightly privileged interface to George 3. A standard High Level Scheduler (with parameters) was provided but an installation could enhance or completely rewrite it if the local policy required scheduling action not handled by the standard program.

The Low Level Scheduler ran the jobs passed to it by the High Level Scheduler with the object of optimising machine usage. It handled such issues as swapping programs in and out of main memory when they were waiting for something; this was particularly relevant in a MOP environment. A subset of the programs being run by jobs known to the Low Level Scheduler was handed over to Executive in numbered slots.

The Executive time sharer handled a number of programs known to Executive, all of which were in memory. Executive switched between them in accordance with their order in the Executive list and whether they were able to run or were waiting for some resource to become available.

Some later 1900 series processors had hardware paging, based on the concepts developed on the Ferranti Atlas in the early 1960s. These systems ran a modified version of George 3, called George 4. The main differences were in the operation of the low level scheduler which arranged for parts of various programs, according to quotas, being held in memory. George 4 had extra user facilities, especially to allow users to use non-contiguous addresses, this was useful for such scientific operations as inversion of sparse matrices.

6 MOP and Remote Batch Processing

The MOP (Multiple Online Processing) parts of George 3 handled many simultaneous users developing and running programs from teletype terminals (visual display units did not become widely available until the 1970s which is beyond the scope of this paper). Users logged in to the system and quoted passwords, this being an alternative to the JOB and RUNJOB commands used to run batch, or background, programs. MOP users had access to the same commands and macro facilities as background users. There was also a break in facility which allowed MOP users developing programs to interrupt the program currently running, examine its data or modify it, and then either continue running it or start again. MOP users could also start a background job and then disconnect from it and do something else or log out. They could, if they wished, reconnect to this job from time to time to examine its progress.

There were also facilities for handling remote peripherals via communications lines. A typical case would be a machine with a number of line printers at various locations the requirement being to print the output at a location convenient to the user. This was done by a PROPERTY command which was a general facility to ensure that specific peripherals were allocated to a job. Example properties could be geographic location, special stationery loaded or ability to print fast.

7 Accounting and Budgets

George 3 provided log analysis for each job. Each job had a monitor file and the log analysis program, which had many parameters and could be replaced by a user-written program, calculated the cost of the job. Each user had a budget, which contained a number of budget types such as money, CPU time and number of magnetic tapes. At the end of the job the amount of money and CPU time used was deducted from the user's account. If the user was overdrawn he was not allowed to start another job. Budgets were refreshed by period accounting which calculated the bills per user. CPU time was divided into a number of urgency levels allowing individual some say in deciding which of his jobs were urgent. The magnetic tape budget specified the maximum number of tapes the user could have assigned to his jobs at any one time.

8 George 1 and 2

George 1 ran as a single program under Executive with the privilege of being able to manage a PUC (Program Under Control). Basically it read in a job control tape (or cards) for one job at a time and ran it. This included handling of editing, compiling, monitoring while running and accounting at the end of the job. George 1 was memory resident.

George 2 essentially added a spooling facility. A number of job descriptions and data files were read in initially and stored either on magnetic tape or direct access backing store. George 2 had a number of overlays; there were various versions depending on whether spooling and overlays were on direct access devices or magnetic tapes or a combination of both.

Minimop was a simple controller of a number of online development jobs which ran under Executive. It could be run simultaneously with George 1 or 2 on machines with insufficient configuration to run George 3. this combination was also used before the release of George 3 in April 1969.

3.5 ICT/ICL 1900 Software - Two views from the Board

3.5.1 – The ICT Software in 1967

Extract from I.C.T. Annual Report and Accounts 1967

"We have continued to maintain a commanding lead in the provision on time of a wide range of sophisticated computer software, and our software-producing organization is now the largest outside America. The basic design of the 1900 Series greatly simplifies the production of software which is efficient, reliable and fully interchangeable between any machine in the range. It is this feature which makes a 1900 Series machine one of the easiest computer systems to install and operate. It also illustrates the fact that I.C.T. is marketing the most comprehensive range of fully-compatible computers in the world.

The software library now available to our customers contains more than 400 major packages and 1000 subsidiary programs, with a total number of "words", or computer instruction steps, of over three million. The library includes important packages ; as **PERT**—Program Evaluation Review Technique—which is a network planning technique for the control of new projects. Amongst many applications, 1900 Series PERT was used to progress the fitting of the main engines of the new Cunard liner, Queen Elizabeth II; and Eldo, the European rocket launching organisation, is using the I.C.T. package to plan and coordinate the development and construction of space launching vehicles. Our lead in the use of the technique was underlined by the successful conference for network planning users which we organised last June; this was attended by over eleven hundred delegates from many parts of the world. Another major 1900 Series software package recently released, is **PROMPT**—Production Reviewing, Organising and Monitoring of Performance Techniques—which consists of four separate sets of interlocking computer programs enabling users to build up computer-based production control systems

On the software development side, substantial progress has been made in implementing highly sophisticated operating systems for the larger computers in the 1900 Series including the 1906A. Operating systems enable a computer to organise and schedule its own work with minimum human intervention, and are an essential feature of any large multi-access computer system. I.C.T. expects, by the middle of 1968, to make multi-access software generally available to its customers, as distinct from the provision of specialized systems for particular applications; it should thus be among the first in the world, and certainly the first European computer manufacturer, to do so."

3.5.2 – The ICL 1900 Software Development in 1969– After the 1968 merger.

Extract from ICL (Holdings) Annual Report and Accounts 1969

Software Development and Production

ICL has more than two thousand staff engaged in the task of developing and producing software. Expenditure on software development alone is now about equal to that for hardware development. The ICL team, the largest software organisation outside the USA, has brought together many of those who have been responsible over the years for major developments in this field. The strength of this organisation reflects the growing importance of software as part of total computer systems.

A major addition to 1900 Series software was made half-way through the year with the release of the **GEORGE 3** operating system. This is currently the most sophisticated of the GEORGE series of operating systems for large 1900 computers. Further marks of the GEORGE 3 system are being developed to enable users to obtain still greater advances in operating efficiency.

The J-level operating system for System 4 computers is now proven and is demonstrating the soundness of the philosophy underlying System 4 software. The Edinburgh Multi-Access Project, an advanced System 4 operating system being developed in collaboration with Edinburgh University, has made encouraging progress and will be put into use on an experimental basis early in 1970, This year will also see the release of two further System 4 operating systems, MULTIJOB and R, the latter being designed specifically for the powerful 4/70 computer.

ICL not only provides basic and applications software for its computers. Its User Programming Service produces and sells programs to customers on a contract basis. This is now a rapidly expanding part of ICL's business.

3.6 - ICT/ICL 1900 Software -

To be provided