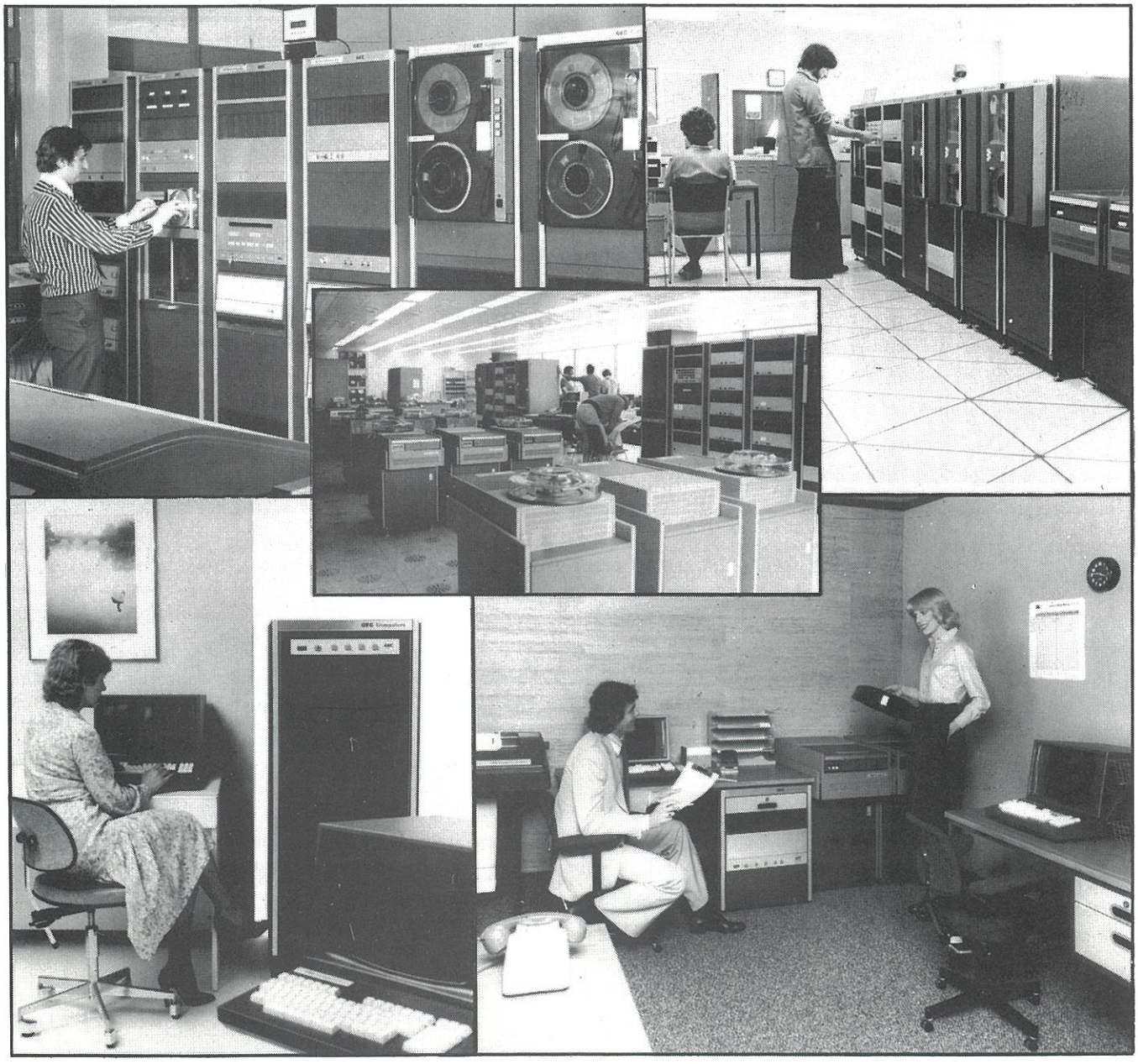
D0534

GEC Computers Limited

GEC 4000 Series computer systems

An introduction



The GEC 4150 central processor (top) with 8" disc drive unit

Introduction

The GEC 4000 Series of computers is composed of a range of powerful 32-bit processors – GEC 4150, 4160, 4180 and 4190. All are suitable for use in applications which present both organisational and computational problems of some magnitude. Their multi-programming hardware, software and sophisticated order structure can greatly simplify the design and implementation of such systems.

Series architecture

Any real-time system can be considered to be composed of a number of 'tasks', which are jobs to be done. These arise asynchronously and in parallel, and at any time there will be a number of tasks being processed, generally at different stages of completion. Each task will require a number of functional programs, or 'processes', to carry out its work, and each of these processes could do work for a number of tasks. Hence, the system as a whole comprises a network of processes, with different tasks taking different paths through the network.

On the GEC 4000 Series, each process is written in 'single thread' form, so that it is only carrying out work for one task at any one time. Communication between processes is by 'inter-process messages', which define the parameters of tasks, and are passed through the appropriate sequence of processes to complete the processing.

A process is activated, or given work to do, by receiving a message from some other process or the outside world. It then analyses the message parameters to determine which operations should be performed, carries out the necessary processing, and finally generates a new message which it sends on to the next process in the network, thus initiating the next stage in the processing of the task. Some processes can communicate directly with the host environment, through sensors, controls, or with human operators.

Processes are multi-programmed on the GEC 4000 Series central processor in such a way that at any time the most urgent process which has useful work to perform is in control of the central processor unit (CPU).

Three features are needed to provide a system structured in the way described above.

First - Protection. Each process has allocated to it certain system resources. It is necessary to ensure that a process can only access its own resources and is precluded from unauthorised access to any others.

Second - Communication. Processes need to be able to communicate with each other by passing messages over well-defined interfaces, or 'routes'. A process can only send messages to, or receive messages from, other processes if it has the necessary route capabilities.

Third - Scheduling. A mechanism must be provided to ensure that control of the CPU is given to the process of highest priority which has useful work to perform.

In most computer systems, these three requirements are implemented in software. On the GEC 4000 Series, however, they are implemented in hardware, and this hardware is called the Nucleus.

Registers and instructions

The instruction set of GEC 4000 central processors provides an extensive repertoire of operations on a large number of data types. Operations on single bytes, variable length strings of bytes, integers of 16 and 32 bits, and single bits within a 16 bit halfword, may all be performed with single instructions. The floating point hardware provides operations on floating point operands of 32 and 64 bits. A wide range of register to register and control operations is also provided.

The main set of *store reference instructions* is provided with a wide variety of modes for addressing operands in store. References to simple variables, record structures, and arrays may all be accomplished with single instructions. An instruction execution rate of up to 1.6 million instructions per second can be achieved with a GEC 4190.

The hardware supports the *multi programming* of up to 256 independently scheduled processes; a set of hardware base and range registers (the segment registers) protect the store occupied by the component segments of these processes from unauthorised access by another process, whilst allowing the sharing of code and data between processes in a flexible and controlled manner.

The *register architecture* of GEC 4000 processors has been carefully chosen to suit the application environments in which the processors may operate. In real-time systems, overheads such as the dumping and reloading of registers must be kept to an absolute minimum without sacrificing the flexibility afforded by a multi-register architecture. These conflicting requirements and the need to define the uses of the registers, so as to maintain interface compatibility between modules, have been taken into account in the design of all machines in the range.

The eight program accessible registers are:-

A — The main accumulator of 32 bits.

B — The main floating point accumulator, also of 32 bits. When double precision is required the A register is used as a rightward extension of B.

X — The index register of 16 bits. As well as being used as an index register for array accessing, it can also be used as a secondary accumulator.

S — The sequence register of 16 bits, which holds the address of the next instruction in sequence to be obeyed.

L — The local workspace pointer of 16 bits which holds the base address of the area of data. This register is normally manipulated by Nucleus for each program unit which is entered. This unit is called a chapter.

Y & Z — The base registers, each of 16 bits, which hold the base address of areas of store. These registers are manipulated from program to address 'record' structures.

C — The conditions register which contains a number of flags which indicate the result of the last operation performed.

The main set of instructions provides operations between these registers and operands in main store; operations on practically all the data types likely to be encountered in real-time operations may be performed with single instructions. For example, the following operations are provided between the accumulator A (and its leftward extension B where appropriate) and the operand:-

Arithmetic operations on 8 bit bytes, 16 bit halfwords and 32 bit fullwords.

Logical operations on 8 bit bytes, 16 bit halfwords and 32 bit fullwords.

Single precision operations in floating point arithmetic, 32 bit.

Double precision operations in floating point arithmetic, 64 bit.

Bit manipulation operations of a single bit in a 16 bit halfword.

Byte string manipulation on variable length byte strings.

Register-store instructions have the following address formats:

A1 Global, direct, non-indexed

A2 Base relative, direct, non-indexed

A3 Global, indirect, indexed

A4 Base relative, indirect, indexed

A5 Global/base relative, direct, indexed

A6 Indirect

A7 Indirect and 16 bit index register

A8 Indirect and 32 bit index from store relative to zero base

A9 Indirect and 32 bit index from store relative to base register

A6 to A9 are one fullword (32 bits) long, and may be aligned on any halfword boundary. A1 to A5 are all one halfword in length.

Addressing facilities

Each instruction within a GEC 4000 process operates in terms of virtual addresses, and Nucleus performs the address mapping to main store addresses at run time. The current virtual address space is occupied by four segments, each of up to 16k bytes, together with a paged address space of up to 16M bytes.

In order to address a new segment or page the reference to it has to be 'fetched' by Nucleus. If the segment being transferred to the virtual store is already in main store this fetch takes only microseconds. Nucleus maintains a table describing the length and position in main store of each segment — it uses this table to set up its registers so that it can perform the run-time checks on address violations. Segments are therefore the unit of address protection; they are also subject to 'access protection'. A separate access protection is provided for each process using a segment.

A system can have up to 16384 segments.

Input/output

Input/output for GEC 4000 systems is provided either by an autonomous input/output multiplexer channel or by one or more input/output processors (IOPs), which are controlled by the CPU using a command interface.

The autonomous multiplexer channel is an integral part of GEC 4150 and 4160 processors, providing the basic input/output capability with a general purpose databus peripheral interface called the Normal Interface. This built-in multiplexer can interleave up to 256 concurrent transfers, with the system software being involved only at the initiation and termination of the transfer of blocks of data.

An input/output processor is standard on GEC 4180 and 4190 systems. This provides the same Normal Interface but is capable of much higher throughput. Additional input/output processors may be added to deal with devices which require very high data rates.

Software

The machine language for GEC 4000 Series computers is a high level assembler, Babbage. The design of this language, which was undertaken in parallel with the design of the machine order code, has resulted in a language which combines the ease and clarity of expression of a high level language with the precise definition of an assembly language. User languages include Fortran 77, Coral 66, Basic, RPG 2, APL, Algol 60, Cobol and Pascal.

GEC 4000 operating system software has been oriented toward the needs of real-time systems. It emphasises support for, rather than control over, the application system, whilst Nucleus maintains secure protection between individual processes within the system. Special attention has been given to assisting the design and implementation of real-time systems.

Nucleus functions

• Store protection

The store protection of GEC 4000 Series computers uses hardware base and range or segment registers. In this system, a virtual address generated by a program is modified by adding to it a base address held in one of the segment registers before transmission to the store.

Simultaneously, the address is checked against the range also held in the segment register to ensure that only the permitted area of store is accessed. GEC 4000 systems provide for the virtual address space of a program to be divided into four segments, each of which is mapped into actual store by one of four hardware segment registers. The size of each segment may be varied between 64 bytes and 16k bytes and each may be positioned anywhere in the actual store. The system permits each segment to be tagged for different types of access and also permits the sharing of segments containing either code or data between programs.

In addition, special extended addressing features are provided to allow efficient access to large data arrays. These are mapped by either the Fortran 77 or Babbage language translators into special 16k byte segments called pages. Pages can be accessed by up to eight additional hardware segment registers each holding base and range information for the most frequently accessed data pages of the current process. The combination of all the segments normally executed in the current virtual address space and the extended paged address space (PAS) constitute the total program address space. This allows processes up to 16M bytes in size to be loaded and executed under the control of Nucleus.

• Inter-process communication

Asynchronous processes in a GEC 4000 Series computer normally communicate by sending messages to each other.

In order to send a message, a process must place a number of parameters in the accumulator and in other registers, and then obey a special instruction which initiates the message passing mechanism of Nucleus. These messages have three components:

(a) Routing information

(b) Four parameters

(c) A data segment (optional)

A process may transmit a message over any of a number of routes. Routes are normally 'paired' to allow bidirectional communication. In this case the entry number is used as a

return route number. The format and permissible routing of messages is predetermined by the system manager. Nucleus checks the validity of routing and format against tables held in an area of main store that is not directly accessible to any normal processes. The route table also contains system information, such as the type of buffer to be used, and its address.

When a process requests transmission of a message it is allowed to specify its next state (not all states are allowed), and whether to make the segment read-only to the destination process. This enables many useful double operations to be performed as a single instruction. For instance it is possible to send a message and wait for a reply.

• Peripheral commands

It is one of the most significant features of the system that programs communicate with their peripherals by passing messages to them. Such messages are very similar to interprocess messages. This enables real-time programs, which must communicate directly with their own peripherals, to be tested in the absence of those peripherals. The route they believe to connect to their peripheral may in fact connect to another process which can simulate the action of the peripheral. Similarly, interrupts are mechanised by allowing peripherals to send messages back to the process that initiated their operation.

A typical peripheral command message contains the following items:

(a) Routing information

(b) Start address

(c) Count

(d) Command

The same basic mechanism that Nucleus uses to control segmentation and message passing also controls input/output. The segmentation mechanism verifies that the store areas allocated for an autonomous data transfer via the Normal Interface are available to the process initiating the transfer. Simultaneously, the message passing mechanism verifies that the peripheral is allocated to the process. 'Interrupts' are treated as messages originating with the peripheral and vectored to pre-determined processes.

• Semaphores

While the system is strongly based on the message passing mechanism for dealing with asynchronism it would not be realistic to assume that this will be adequate in all cases. For instance, even when implemented by hardware, this mechanism carries a heavier overhead than hardware semaphores (current semaphore systems are software mechanised and carry very heavy overheads, but are still tolerable). In some simple cases it may be considered that the greater safety of the message passing mechanism is not justified against the heavier overhead.

Therefore GEC 4000 computers provide binary semaphores, held as ordinary data items in the data space (presumed shared) of the processes. The semaphore operations are invoked by two hardware instructions, CLAIM and RELEASE. The operation of suspending a process when it becomes HELD on a semaphore is performed automatically by Nucleus and is invisible to that process, as is the rescheduling invoked should the process release a semaphore that is awaited by others. A CONDITIONAL CLAIM instruction is provided for those circumstances in which a process must avoid suspension.

• Process scheduling

The Nucleus scheduling system is based on a simple fixed priority sequence for the various processes, the scheduler scanning a list of processes to discover which one to run. This system lends itself to hardware implementation, thus considerably reducing the overheads on the system, and allows rapid response to demands from the real-time environment.

The operation of the scheduler is quite straightforward. It maintains a list of processes in order of priority, and records for each process its present state. The scheduler scans these processes in sequence, looking for the first process which may be run. If during the sequential scan it encounters a process which is waiting for some other process, then the sequential scan is suspended and the process causing the waiting is examined to determine whether it can be run. This algorithm ensures that when a high priority process is blocked by a process of lower priority, then the lower priority process is encouraged to release the blockage.

• Nucleus error reporting

When a violation is detected by Nucleus, it itself sends a message giving details of the violation to a routine in the executive software.

The error conditions trapped by Nucleus are hardware errors, system errors, programming errors and segment breaks.

(a)	Hardware errors	—	These include Store parity error, Store timeout, Command interface, timeout, Power failure.
(b)	System errors	—	Protection violation, Queue counting error, Stimulus error.
(c)	Programming errors	—	These include Store protection violation, Illegal instruction traps, Explicit traps, Nucleus errors.
(d)	Segment breaks	—	Nucleus attempts to load a segment into virtual store which is not in main store.

• Special tasks

When Nucleus discovers that an executive task beyond its scope is necessary, it sends a message to a routine in the executive software. A typical task in this category is the bringing into main store of a process on backing store for which there are messages awaiting attention. The use of the message passing and priority mechanisms ensures that all tasks are given appropriate precedence and that, just because it originates in the executive, a report of a violation by a background program is not given priority over the running of an urgent real-time process.

Associated data sheets

Detailed data sheets are available describing the processors, stores, peripherals and software for the GEC 4000 Series. A summary of the items available may be found in the following data sheets:

GEC 4000 Series hardware quick reference list
OS 4000 software quick reference list
Input/output peripherals quick reference list

GEC 4000 Series comparison

		GEC 4150	GEC 4160	GEC 4180	GEC 4190
Minimum store (Mb)		¼	¼	1	1
Maximum store (Mb)		1	1	4	8
Mill width (bits)		32	32	64	64
Store highway (bits)		16	16	32	32
I/O width (bits)		16	16	16	16
Virtual address space (Mb)		16	16	16	16
System address space (Mb)		256	256	256	256
Typical instruction times - (microseconds)					
add	32 + 32 ⟶ 32	3.6	3.6	1.7	0.7
multiply	32 x 32 ⟶ 64	10.1	10.1	6.8	2.2
divide	64 ÷ 32 ⟶ 32	12.6	12.6	7.6	7.3
branch		1.5	1.5	1.0	0.6
floating point	add	7.2	7.2	2.7	2.2
(single	multiply	10.9	10.9	6.1	2.7
precision)	divide	12.9	12.9	7.2	6.8