FROM THE EDITOR

I found it disappointing that a computer system, the design features of which contributed so heavily to making timesharing practical, the Gamma 60, was not represented in the book which is a candidate to be the classic in computer systems architecture — Computer Structures: Readings and Examples, by C. Gordon Bell and Allen Newell. Upon my query, Professor Bell said that they just could not find anything in English that was sufficiently descriptive of the real functioning of the Gamma 60; he, too, regretted the omission.

Dr. Fred Brooks, who reviewed this book for Computing Reviews, also noted that "the selection heavily favors American machines . . . only eight European machines are included, in spite of the flow of important new concepts from Europe . . ."

M. Bataille, of the original team, has given us a new description, though there has not been sufficient time to put it into the Processor-Memory-Switch (PMS) and Instruction Set Processor (ISP) notation of Bell and Newell. Let that be a challenge to some candidate for a degree in computer science. Surely with ten systems still operating this will not be difficult.

the GAMMA 60

the computer that was ahead of its time

M. Bataille

Prior to 1960 the Compagnie des Machines Bull (now Honeywell Bull) delivered the first large computer system with an architecture designed for multiprogramming. Many unique features of the Gamma 60 were forerunners of present system architecture concepts. This article revisits these concepts.

INTRODUCTION

The Compagnie des Machines Bull designed a data processing system, named Gamma 60, which was publicly described in the US in 1958 May [5]. Twenty Gamma 60 systems were produced, ten of which are still in operation. For this reason we use present tense in the description. The systems design was revolutionary for its time, it being the first architecture specifically designed for parallel and multiprogramming. It might be well to remember that when the Gamma 60 was designed, UNIVAC I, the IBM 705, and the RCA 301 were names of computers known to the public. Many of the unique features of the Gamma 60 are still in use, indicating the extensive influence of its innovation. To list them:

- The interrupt instruction, which delimited parallel asynchronous sequences, was the forerunner of present-day processes. The number of processes was, however, defined when the program was coded, rather than being determined dynamically.
- The combination of interrupt instructions, Program and Data Distributors, and Control Return Chain is the forerunner of present-day process management and synchronization, using semaphores. Interrupt instructions specifying parallel paths for program concurrency were later named FORK and JOIN by Conway [18].
- The Control Return Chain is the parent of today's queuing mechanisms.
- The virtual element protecting a subroutine is a parent of the semaphore.
- The distribution of functions between distributors, channels, and functional units is accepted practice in present-day systems (i.e., the IOC function in multiprocessor systems).
- These technological features enabled the first multiprogramming to function in a software environment. However, there was no hardware protection of memory.
- The "status catena" is the parent of the "status word", which did not exist in 1957-vintage computers.
- The use of phase modulation instead of non-return-to-zero recording for magnetic tape was an innovation in its day, and was not used in other computers until several years later.
- The Gamma 60 character set was remarkably similar to the present ASCII/ISO code. It was by far the closest to it of the 65 computer character sets surveyed in 1960. The blank, alphabet, and digits are identical in the 4 low-order bits.

GENERAL ORGANIZATION

Figure 1 shows the organization of the Gamma 60. Its three main sections, which are interconnected in pairs, are the Central Memory (CM), the Central Processor (CP), and the Data Distribution and Collection Channels (DDC and DCC) which link the operational units. The interconnections shown as double lines are those used by the catenae. (Catena ("chain") is the 24-bit *word* or *byte* — the data quantum.) The DDC and DCC are focal points of the system, through which all data must pass. The various input, output, and processing units receive data via the DDC and send it via the DCC.

As each unit transfers data individually at its own rate, a central component known as the Data Distributor (DD) controls communication with the central memory so that each operational unit may request data in turn. By organizing a priority chain, the Data Distributor regulates such communications and controls channel opening (shown by the single lines in Fig. 1; the circles may be thought of as valves to open and close).

Operational units are connected to the channels via 6 special circuits, as follows:

class 0 — console typewriter, logical unit (binary computation)
class 1 — general comparator, transcoder, and arithmetic unit
class 2 — magnetic drum and magnetic tape units (2 circuits)

class 3 — unit record multiplexor, card readers, punches, and printers

class 4 — paper tape readers and punches, typewriters, etc.

Connection of several teletypes, via a special multiplexor, for remote direct access was attempted and found possible.

The central processor is connected to the central memory, from which it requests program instructions to carry out. It is controlled by the data distributor in doing this, like any other unit. The central processor is also connected to the channels, through which the operational units request instructions and receive detailed commands.

Before discussing these data transfers, known as the central processor's dialog with the operational units, we must first cover the simultaneous operation capabilities of the Gamma 60. Simultaneity is the Gamma 60's main feature; although there were at design time other systems which could handle a main program while data is being input and output, the Gamma 60 is the first electronic computer to handle simultaneous and independent operation of all the components included in its organization. This not only optimizes the operating time of a program, but it means that a single machine can work on several separate problems with different programs being executed simultaneously. Optimization is brought about by maximum operation of the central memory, which is the heart of the system.

The operational units execute their assigned tasks in the same amount of time as if they were operating alone, since central memory operation is much faster than any of these units. Necessary control is provided by the data distributor. Operating simultaneity is absolute when the units are not transferring data to or from the central memory, but is less so at the level of memory cycle time (10 μs) and can be considered as a sort of multiplex.

CENTRAL PROCESSOR DIALOG WITH UNITS

To illustrate the dialog between the CP and the units, let us consider the operation of the magnetic tape unit:

- The program requests reading of data (a block) from tape. A program interrupt instruction is sent to the tape unit to start its operation. (The various types of instructions are discussed in the next Section).
- The tape unit replies with an instruction transfer request (ITR). This request is queued, since the program distributor (PD) component of the central processor (in charge of processing instructions) does not remain idle while waiting for the reply from the tape unit, but continues handling other ITRs that may have been received.
- When our unit's turn comes, a signal is sent to it. It answers by giving its number, the result of the operations carried out earlier, etc. (status catena).
- The program distributor then looks for the line number (or address) at which the program calling the tape unit had left off. This address is stored in a register which is specific to each unit (but located in the central processor), called the Program Address Register (PAR). The instruction at this address in the central memory is processed first, and processing continues sequentially until the DP reads a command instruction.
- This command (in this case a "read") is the only instruction sent to the tape unit. The program distributor goes on to another task while the tape unit is being started up. As it reads data from the tape, it gathers the data together by catenae. Each complete catena stored is prepared for transfer to the central memory.
- To do so, the PD sends a data transfer request (DTR) which joins the data distributor queue. When its turn comes, it is so informed by the data distributor, which requests that the corresponding channel be opened.
- The tape unit then sends its message, which includes, among other details, its identification number (other types of units may include additional information in this message).
- Using this information, the distributor seeks the address in the central memory to which the catena is to be transferred. This address is found in the Data Address Register (DAR), which, like the PAR, is also specific to the unit and is located in the central processor. After incrementing or decrementing this address according to indications supplied by the unit, it stores the updated address in the DAR to be used for the next catena. It then goes on to handle queued data transfer requests.

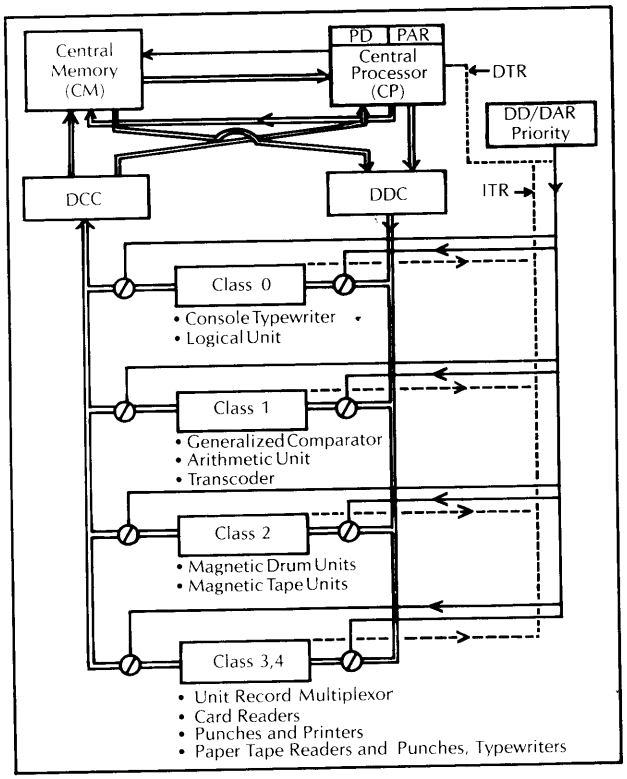

FIGURE 1 Gamma 60 Block Diagram

- The reading process continues until the entire block (i.e., all data between the two start-stop gaps) has been read. At this point, it requests a new order by sending a new ITR. It may now receive another "read" command or an order to halt (if this phase of the program no longer requires the use of this tape unit).

THE INSTRUCTION CODE

Before we can discuss what occurs in the central processor between the sending of commands to the units, we must have some idea of the way programs are written, and of the structure of the instructions.

The Gamma 60 is not classifiable as being an n-address instruction machine, where n is fixed. Its instructions may contain anywhere from 1 to 4 addresses. They consist of a variable number of catenae, each catena containing a "canonical" instruction. The complete (variable length) instruction is processed without interruption. The simultaneity on processing program sequences that permits multiprogramming is obtained by the more or less random overlapping of processing of complete instructions.

When the PD (which processes instructions) has to communicate with the central memory, it transmits a DTR as would the other units that are generally communicating with the CM at the same time.

Each canonical instruction (1 catena) contains a 15-bit address identifying a central memory address, a variable, a unit number, or a length, etc. It also contains a code which identifies the instruction as belonging to one of four major categories:

Address	A
Branch	B
Interrupt	C
Command	D

The instructions in these categories will be discussed in logical order, not alphabetic.

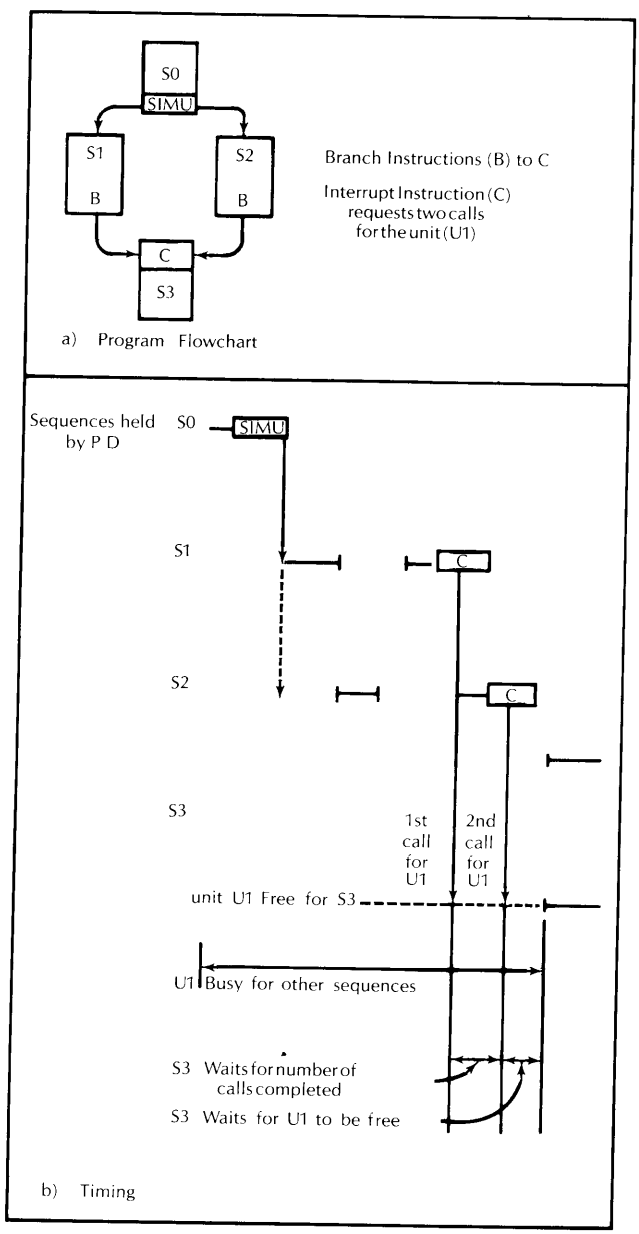

FIGURE 2 Use of an Interrupt Instruction to Join Simultaneous Sequences

- Interrupt Instruction

The interrupt instruction is used to start an operational unit. The number of the unit is included in the instruction. The complex effects of this instruction will be discussed later, but the one that interests us at the moment is that of loading into the program address register (PAR) of the unit the address following its own address. For example, if the interrupt was at address 748 and indicated the arithmetic unit, the address 749 will be loaded into the PAR register for the arithmetic unit (provided this unit is not otherwise occupied). The instructions executed for this arithmetic unit will be those contained in addresses 749, 750, 751, etc. In addition, the "status busy" symbol will be placed into another position of the PAR.

Since simultaneous sequences are possible in the Gamma 60, the interrupt instruction is also used to combine sequences that were started up simultaneously (by means of SIMU instructions, described later). These sequences will all be executed individually, but they must all be completed before the processing of the problem can be continued. In this case, each of the sequences returns control (by a branch) to the same instruction C (Fig. 2). This instruction not only indicates the unit (U1) which is to process the data produced by the various sequences, but also indicates the number of calls it must receive before it is ready to continue. (The programmer must leave one catena free or blank following each interrupt instruction, which is mainly reserved for the control return chain, but also counts calls that have been received. This is the mechanism which provides processing simultaneity, a specific Gamma 60 characteristic.)

- Address Instruction

  By operating on the DAR, this instruction prepares the address that will be used as the starting point for data transfers between the central memory and the operational unit. The address instruction may fill the register, change its content by addition or deletion, or transfer the content to a central memory register for future processing. In a complete instruction, there may be as many address instructions as there are DAR registers for the operating unit.

- Command Instruction

  These are the only instructions sent to the operational units, and start their operation. The units may recognize a different number of command instructions, depending upon the complexity of the unit. For example, the arithmetic unit recognizes 13 commands, such as addition, division, roundoff, etc.

- Branch Instruction

  These instructions are used to skip sequences, either explicitly or on condition. A conditioned branch takes the path indicated by the match between the branch number and an index located in the unit status catena.

  One special instruction is classified among the branches, but has features very close to interrupt instructions. This is the simultaneous branch instruction known as SIMU, which takes advantage of the Gamma 60's multiprogramming capability by starting up a new sequence simultaneously with the sequence that would normally come next. This operation will be examined more closely in the discussion on the control return chain principle.

- Addressing Schemes

  It is possible to operate upon the addresses within the canonical instructions by means of the differential or substitution functions, leading to relative and indirect addressing correspondingly.

SIMULTANEITY

- Waiting for a Unit

  A program sequence requests use of a unit by including its number in an interrupt instruction. The requested unit may be free, or it may be busy processing for another sequence, in which case the sequence requesting the busy unit must wait until it becomes free. The PD must not, however, be blocked from processing other executable sequences without having to queue. The interrupted sequence is therefore placed into reserve, to free the PD. This establishes a queue known as the control return chain (Fig. 3). Any sequences waiting for the same operational unit are added to this queue (as at a ticket window which can handle only one person at a time). When a unit is free to accept work it requests that the PD send the first sequence in the queue for processing. The control return chain may be as long as may occur; it is built in FIFO (first in, first out) order.

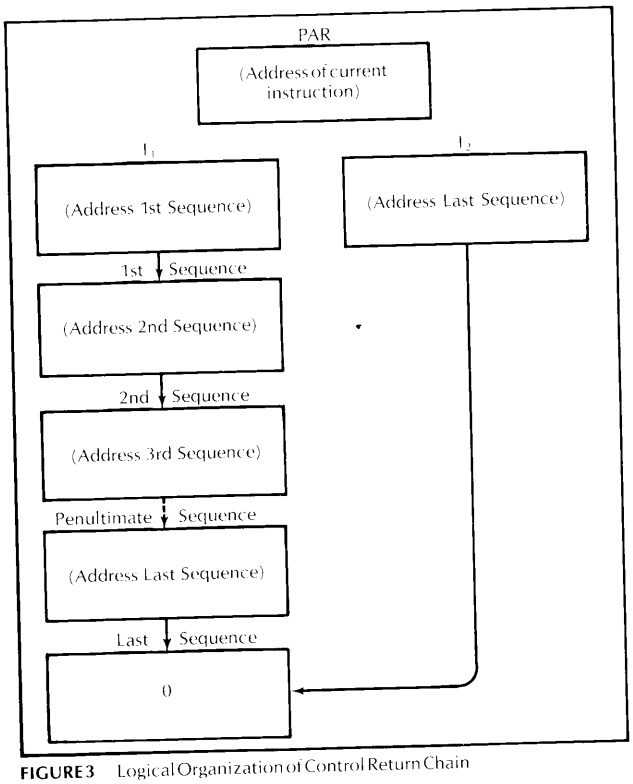

FIGURE 3 Logical Organization of Control Return Chain

- Special Logical Organization

  Each unit has three registers allocated to it: the current program address register (PAR) and I_1 and I_2, which are located in the reserved part of the central memory (addresses 0 to 127) and are used for the operation of the control return chain. Queued sequences, waiting to be processed, are identified by the address or line number of the catena following the interrupt instruction. Register I_1 stores the address for the first queued sequence (i.e., the first link in the chain) and I_2 stores the address of the last sequence.

  Thus I_2 is used to build up the chain by adding another link. The intermediate links connect with each other via the catena left empty by the programmer after each interrupt instruction. This catena is used (among other functions) to contain the address of the empty catena following the interrupt of the next sequence in the chain.

- Simultaneous Programs

  The SIMU instruction, first appearing in the Gamma 60, is set off by an interrupt instruction. It

is a canonical instruction used to divide a single program into two sequences, each to be run separately. This is done through a control return chain of the PD. The SIMU calls the instruction at the address given, but keeps the remainder of the current program in store by taking advantage of the PD's capability of creating its own control return chain. In this way the programmer can run the various segments of his problem just as in the different branches of his flowchart (Fig. 2). By using successive SIMU it is possible to launch any number of simultaneities.

- Virtual Units Protection of Subroutines

  Program execution simultaneity could cause difficulties in using standard routines, if there were no protection features. As an example, take two programs each involving computation of the sine. The program for sine computation would not be duplicated, but performed by a subroutine called by a branch instruction, and control returned to the main program after computation.

  No other program must be allowed to call the sine subroutine while it is being executed, because it would be impossible to return control to the first program (Note — pure procedure techniques were not used at the time of this design). To inhibit simultaneous calling of a subroutine, it is treated as though it were an operational unit, which is why it is called a virtual unit. Its own queue can be set up. It is protected by an interrupt instruction which starts the virtual unit and signals "red light" or busy. On termination, another interrupt instruction frees the virtual unit and changes the signal to "green" so that the queuing program can allow the next program to call it.

  This process is identical to the control return chain process for operational units, even to the I_1 and I_2 registers and the control circuits.

BRIEF SUMMARY OF PERIPHERALS AND TECHNOLOGY

- The generalized comparator (class 1) compares binary data in strings from 1 to 255 catenae in length.
- The transcoder (class 1) performs code conversion, to and from the internal character code and the I/O device encodings, and edits for output. Two special core matrices are used.
- The arithmetic unit (class 1) operates in binary coded decimal format, with 4-bit digits. Self-checking modulo 7.
- Magnetic drums (class 2) store 25,600 addressable catenae, 200 per track. At 3000 rpm, transfer rate is 100 μs per catena, with average access of 10 ms.
- Magnetic tapes are recorded in phase modulation, at 280 bpi. The read head checks after write (write forward, read both directions). Mylar tape length is 1100 m, width is 12.7 mm (0.5 in), 7 tracks plus a synchronization track. Maximum transfer at 1.9 m/s (75 ips) and 15 KC is 3750 catena/s. Maximum capacity is 38,000,000 bits. Switching between 12 tape handlers and controller is carried out directly at read-head level.
- High-speed memory is in 1 to 8 blocks of 4096 catenae (of 24 bits), with a 10 μs cycle. Addresses 0 through 127 reserved for special purposes.
- Both card reader and punch operate at 300 cards/min.
- Wheel printer operates at 300 lpm, with 120 positions of 60 characters each. Echo check.

ANNOTATED BIBLIOGRAPHY

Prof. Saul Gorn, in his introductory speech for the sessions on Common Symbolic Lanquage for Computers, at ICIP, 1959 June:

> "If we can extrapolate the trends in the development of programming languages, I would say that the following is our goal . . . It should be capable of expression in parallel, in order to match the truly parallel machines we can expect as the next generation, and of which the Gamma 60 is a prototype."

1. S. W. Dunwell, "Design Objectives for the IBM STRETCH Computer," *Proc. Eastern Joint Computer Conference,* 20-22 (1956 Dec 10-12).

   The parallelism specified here is the concurrent operation of the I/O, editing of I/O data, and arithmetic. The technical exactitude might be viewed by Dunwell's statement about input-ouput that "interpretation of speech and handwritten data are recognized as being less immediately available, but it is assumed that they will become possible within the life span of the Stretch system."

2. J. P. Eckert, "UNIVAC-Larc, the Next Step in Computer Design," *Proc. Eastern Joint Computer Conference,* 16-20 (1956 Dec 10-12).

   Larc parallelism was essentially in the same areas as Stretch, with the additional possibility of a second arithmetic processor.

3. Gamma 60. Firmenschrift der BULL-Exacta, Köln, 1957.

   Reference and described in K. Prause, "Elektronische Rechenanlagen 1957", (in Ger.) Fachgebiete der Technik in Jahresübersichten **100,** No. 16, 701-708 (1958 Jun).

4. S. Gill, "Parallel Programming," *Computer J.,* **1,** No. 1, 2-10 (1958 Apr).

   "Parallel programming will be used in both the Sperry Rand Larc and the I.B.M. Stretch . . . Time-sharing is not an entirely new idea." The Gamma 60 was not mentioned.

5. P. Dreyfus, "System Design of the Gamma 60," *Proc. Western Joint Computer Conference,* 130-133 (1958 May 6-8).

"The Gamma 60 opens a new era in data processing whereby . . . distinct independent problems can be solved simultaneously on a single machine without any previous planning or programming of the possible occurrence of these parallel operations . . . the only common devices shared by all elements are the main memory and transfer buses . . . The data distributor handles the priority problem of time-slot allocation."

Chairman John W. Carr, III, said "I would like to express what I think a lot of people are thinking and that is that the Bull Company is to be congratulated for very imaginative design."

6. "France's Gamma 60, A Step Forward in Data Processing?", *Datamation,* 34-35 (1958 May-Jun).

7. P. Dreyfus, "Programming Design Features of the Gamma 60 Computer," *Proc. Eastern Joint Computer Conference,* 174-181 (1958 Dec 3-5).

8. C. Strachey, "Time Sharing in Large, Fast Computers," *Proc. ICIP,* 336-341 (1959 Jun).

   This is believed to be the first published paper to use the term "time share" in the title, although it is traceable at least back to J. W. Forgie (TX-2 papers in 1957 WJCC).

   In the discussion, E. F. Codd of IBM said this was similar to the Stretch (7030) computer, to be reported at the EJCC in late 1959. Dreyfus said "I disagree on the possibility of handling time sharing by a program. In my opinion, it must be handled by hardware."

9. J. Bosset, "Sur certains aspects de la conception logique du Gamma 60," *Proc. ICIP,* 348-353 (1959 Jun).

10. E. F. Codd, E. S. Lowry, E. McDonough and C. A. Scalzi, "Multiprogramming STRETCH: Feasibility Considerations," *Comm. ACM* **2,** No. 11, 13-17 (1959).

    "Now we discuss how we propose to exploit the built-in logic by programming techniques in order to meet the six requirements for acceptable multiprogramming."

11. N. Lourie, H. Schrimpf, H. Reach and W. Kahn, "Arithmetic and Control Techniques in a Multiprogram Computer," (Honeywell 800), *Proc. Eastern Joint Computer Conference,* 75-81 (1959 Dec 1-3).

    Ed. Note — *This would appear to be the only widely-published paper on the design (or system architecture) of the Honeywell 800. In face of the multitude of papers on Larc and Stretch, we tended at first to be defensive. Then we discovered that 2 Larc systems were installed, the first in 1960 May; 7 Stretch systems were installed, starting 1961 May. The H800 was installed first in 1960 December, and 89 were delivered! End of embarrassment.*

12. L. D. Yarbrough, "Some Thoughts on Parallel Processing," *Comm. ACM* **3,** No. 10, 539 (1960).

13. P. Davous, M. Bataille and Y. Harrand, "le Gamma 60," *Onde Electrique* **40,** No. 405, 889 (1960 Dec).

14. D. E. Kilner, "The Characteristics of Computers of the Second Decade," *Comp. Bull.* **4,** No. 3, 88-97 (1960).

    "It is only in a machine with multiple control units that fully parallel programming is achieved." Referring to the Gamma 60, the only extant example.

15. M. R. Nekora, "Comment on a Paper on Parrallel Processing," *Comm. ACM* **4,** No. 2, 103 (1961).

    He claims that the Gamma 60 does multiprocessing, which Stretch cannot, only multiprogramming.

16. B. L. Ryle, "Multiprogramming Data Processing," *Comm. ACM* **4,** No. 2, 99-101 (1961).

    Compared are Gamma 60 — central memory, multiple control; Honeywell 800 — multiple memory, central control; RW 400 — multiple memory, multiple control.

17. P. Dreyfus, "Programming on a Concurrent Digital Computer," Notes of University of Michigan Engineering Summer Conference on Theory of Computing Machine Design (1961).

18. M. E. Conway, "A Multiprocessor System Design," *Proc. Fall Joint Computer Conference,* 139-146 (1963 Oct).

    Definition of the FORK and JOIN operations. "The fork-join notion has been around for a while. The equivalent of FORK is elsewhere given these names: in the Gamma 60, SIMU."

19. A. J. Critchlow, "Generalized Multiprocessing and Multiprogramming Systems," *Proc. Fall Joint Computer Conference,* 107-126 (1963 Oct).

    This paper has a comprehensive list of references and a bibliography; a scrutiny of these reveals the major difficulty with introduction of systems of this class: the programming knowledge to utilize the advanced hardware design lagged far behind.

SOME FRENCH PATENTS ON THE GAMMA 60

1.180.399 (with supplement 72.485)
1.180.400
1.184.076
1.231.549
1.232.587
1.254.998

Ed. Note — *US readers will note here a partial example of the inverse European usage of period (full stop) to separate thousands (and the comma for the radix point).*