Note on timing simulation of a large asynchronous computer

By R. L. Chew*

An Autocode program was written for a small computer to simulate the overlapping instruction process of the large Atlas 2 computer, for timing purposes as an aid in design work and in preparing tenders. The note describes the techniques used in writing this program and its applications.

An Autocode program has been written for the small Sirius computer to simulate the more important features of the overlapping of instructions on Atlas 2, a large computer designed by Ferranti Limited in conjunction with Cambridge University, the first model of which is now installed at Cambridge University. An urgent need arose for a program to simulate the behaviour of sequences of basic Atlas 2 instructions. This information was required for estimates of times for various documents and tenders, for basic programmers designing the most efficient loops, and for the assessment of the effects of possible improvements in design. Through discussions with design engineers a lattice was drawn to show the main events occurring in the basic circuitry governing the overlapping of instructions.

Fig. 1 is a simplified version of the lattice used for the timing program and gives the configuration of only the main events occurring in the overlapping of three instructions. Events are represented by circles; each arrow vector represents the minimum time which must elapse between two events. Other time delays, not shown, that must precede certain events are "accumulator busy" hold-ups, "index register busy" hold-ups and "core store busy" hold-ups.

An important factor in the overlapping of instructions in Atlas 2 is that the core store is divided into four stacks each of which may be accessed independently; the stacks are interleaved so that any four consecutive locations belong to different stacks, thus setting no effective limit to the rate at which successive instructions may be extracted. Also there are 40 extremely fast-access registers constructed of tunnel diodes. In 32 of these, hereafter referred to as the *slave store*, loops of instructions are automatically stored while they are obeyed, and any loop of less than 64 instructions benefits from this facility. The remaining 8 registers are provided for use as fast working space. These 40 registers are accessed independently of the main core-store stacks. Some of the delay times used in the lattice are variable, depending upon whether or not an instruction is in the slave store or its operand in one of the eight fast registers.

A description of the program follows showing how a sequence of Atlas 2 basic instructions is represented for input to the timing program, and how time is stepped up as the instructions advance through events in the lattice.

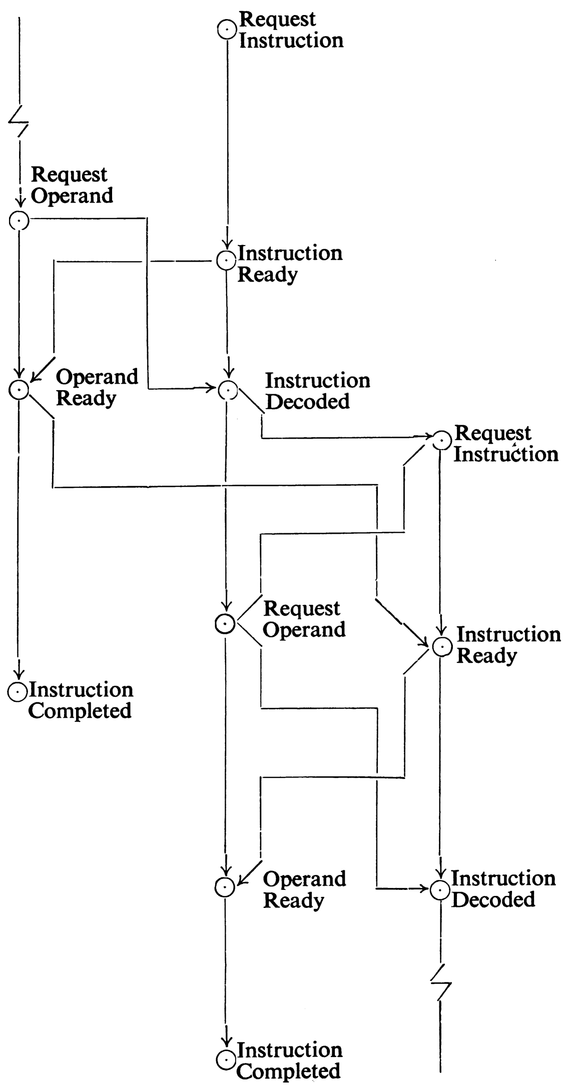

Fig. 1.—Lattice showing configuration of three overlapping instructions

* *I.C.T. Limited*, 21, *Portland Place*, *London W.*1.

First the delay times are read in as parameters; these may easily be varied. A sequence of instructions is then read in as follows:

1. An identifying number.
2. The number of instructions in the sequence.
3. A representation of each instruction.

The identifying number of a sequence will be printed out with the time for that sequence. There is no limit to the number of instructions in one sequence. Each instruction is then represented by five integers indicating:

1. Whether or not it is a jump (this type cannot be overlapped).
2. Where the instruction is stored, i.e. in a stack of core store or in the slave store.
3. The number of index-register modifications, if any.
4. Where the operand is stored, i.e. in a stack or in a fast register.
5. The type of function.

Upon entering the program, the registers holding the times for "accumulator busy" periods and for stacks of "core store busy" are set to zero. Real time T, is set to zero as the first instruction is requested. A delay time, depending upon where the instruction is stored, increments T in nanoseconds to the point "instruction ready". Next, T is incremented by the time appropriate for instruction decoding. Then there is a test for the instruction being a jump, in which case it must be completed before the second instruction is requested. Otherwise, from this point a separate path is followed for each instruction, real time being represented by two variables; at each event it is ensured that the appropriate delays have elapsed. When "instruction decoded" is reached for the second instruction, as before, a test is made for a jump instruction. If this too is not a jump, overlapping for three instructions is now under way.

This process continues until all the instructions in the sequence have been completed. Two more iterations through the sequence are completed to ensure a consistent pattern. The time taken to complete the sequence of instructions is calculated thus: a record of time T stored at the end of the second iteration is subtracted from the value of T at the end of the third iteration, and is printed out in microseconds with the sequence identification number.

The simulation program was written in a period of two months and is now proving very useful in timing loops involved in supervisory control routines; also it is helpful in estimating the effects of different core-store cycle times and of varying accumulator speeds in performing arithmetic operations.

Obituary

It is with deep regret that we announce the death, after a long illness, of Dr. C. B. Haselgrove on 27 May 1964. He had been chairman of the Manchester Branch of the British Computer Society for a few years until prevented by ill health.

Brian Haselgrove was educated at Blundell's and King's College, Cambridge. He was a Research Fellow at King's from 1950 until 1956. He first became interested in computers in the later 1940's. In 1947, while a vacation student, he worked on the design of EDSAC 1 at the University Mathematical Laboratory. Later he joined the senior staff of the Laboratory and played a major part in its teaching work and in the use of the EDSAC.

In 1957 he became Senior Lecturer in Computing in the Mathematics Department at Manchester University. There he helped to start a post-graduate Diploma in Numerical Analysis and also one of the first undergraduate courses in which computer programming was taught. From the earliest, he was aware of the significance of computers to mathematical problems, both pure and applied. He was interested in any branch of the subject which could be tackled by a machine and he did work on number theory, Monte Carlo methods, stellar evolution, radio waves in the ionosphere, non-linear programming and many other topics. He served on the Royal Society's Mathematical Tables committee and in 1960 he published tables of the Riemann Zeta function with J. C. P. Miller.

In his 7 years at Manchester he has fully or partly supervised about a dozen research and diploma students all of whom will remember him with admiration and affection.

The Society extends its sympathy to his widow and son for what, to them even more than to us, is a tragically early loss at the age of 37.