The Manchester University Atlas Operating System Part II: Users' Description

By D. J. Howarth, R. B. Payne and F. H. Sumner

A system of operating Atlas is described from the point of view of the user. The simple additional statements required from the programmer are supplied to the computer rather than a human operator.

The Manchester University Atlas will be used by various departments of the University and others, and will deal with a large quantity and variety of problems, some of which will complete computing in a few seconds. Consequently it is important that the computer operators be relieved of as much work as possible to ensure a smooth flow of work through the computer. This operating system has been devised by Manchester University and Ferranti Ltd., as a result of discussions between Professor T. Kilburn and the authors.

An important feature of the system is that it does not depend critically on how much peripheral equipment is operating; the system can function, though possibly with reduced efficiency, even if no magnetic tape decks are available to it. Normally three tapes are used to implement the system*:

1. the system input tape
2. the system output tape
3. the system dump tape.

Tape 1 buffers input from slow peripherals and also preserves a record of all input from slow peripherals. Tape 2 is similarly used for output and tape 3 has a variety of uses, including a store for programs temporarily held up and, where necessary, as an overflow of tapes 1 and 2.

The layout of all system tapes is the same, sufficient information being recorded to locate any information given the initial address on tape. Facilities are provided for the programmer to use information from these tapes instead of repeating input on slow peripherals; if a long paper tape is read in, it may be used again by referring to its location on a system tape. The location on the system tapes of all slow peripheral input and output dumps, etc., is printed with the programmer's results.

When large amounts of input or output are involved, the programmer may use private magnetic tapes to record the information. This is done by suitable specification in the title.

Titles and Headings

Jobs are initiated on the computer by input of information on a slow peripheral equipment. A job may consist of several sections of information, each preceded by an identifying title. This title, by which input information is known, consists of one line of printing following a heading such as

COMPILER (x)
DATA
JOB

where x may be

INTERMEDIATE INPUT
MERCURY AUTOCODE
FORTRAN
ATLAS AUTOCODE

If the heading of the input information is

COMPILER MERCURY AUTOCODE
(The title)

then the information is a source program in Mercury Autocode language. More generally, the information obeys the rules of Mercury Autocode, and may therefore include data as well as autocode instructions.

If the heading of the input information is

DATA
(The title)

then the information following is data to be read by a program during execution, and which obeys no rules known to the system.

The heading "compiler" does not itself initiate the appropriate compiling, which is only commenced when a "job" heading is read. If the heading is

JOB
(The title)

then the information following is the job description. In general this information is optional. It is terminated by an end-of-tape marker, in the case of a separate steering tape, or by "compiler" or "data," in which case the title is not repeated. The "job" heading will normally precede the source program tape as follows:

JOB
(The title)
COMPILER MERCURY AUTOCODE

then the source program itself.

Further optional information may be included in the job description such as:

* See the paper by Kilburn, Howarth, Payne and Sumner on p. 222 of this issue.

Downloaded from https://academic.oup.com/comjnl/article/4/3/226/380540 by guest on 13 March 2024

1. data and program tapes (input)
2. output equipments used
3. magnetic tapes
4. store required, computing time, etc.

These sections of the job description are described below.

Job Description—Input

A program reads in data by means of instructions which are effectively "read next character/string of characters from data tape *n*" where *n* is a decimal integer. (In this context, "data tapes" are intended to include stacks of cards.) The programmer's number of the data tape is specified in the "input" section of the job description. This section begins with the word

INPUT

and is followed by a list of the titles of data tapes used in this job, each preceded by the programmer's number, *n*, e.g.

INPUT
1 (the title of data 1)
2 (the title of data 2)

where there are two data tapes known by the programmer's numbers 1 and 2. These may have been read into the machine on the same input equipment as the "job" tape, either before or after it, or on other input equipments. The programmer's number, $n = 0$, is reserved for the program itself (and may be used in the program to read in data which follows the program as part of the same tape). A separate steering tape might be

JOB
(the title)
INPUT
1 (the title of data 1)
0 (the title of the program tape)

In this case, the name of the compiler to be used is written at the head of the program tape. When the "job" heading is on the beginning of a data tape, the "input" section of the job description must include

SELF = (*n*)

where *n* is the programmer's number by which this data is known within the program.

If the input section of the job description is omitted, it is taken as if

INPUT
SELF = 0

were included, and the program following is compiled and executed.

Since all input is automatically copied to the system input magnetic tape, a programmer may read his tape in again, direct from this input tape (e.g. to make a correction). To do this, in the "input" section of his job description he writes

TAPE (*a*)/(*b*)/(*c*)
(*n*) (the title of his input)

where *a* is the system tape "number,"
b is the number of the 512-word block of tape, and
c is the line within the tape block where his input starts.

His title is, of course, written on the tape at this point, but the title is specified again as a check.

Job Description—Output

A program puts out results by means of instructions which are effectively "print next character or string of characters on output *n*," where *n* is a decimal integer. The output equipments are specified in the "output" section of the job description. This section begins with the word

OUTPUT

and is followed by a list of the output mechanisms used in this job, each preceded by the programmer's number, *n*, e.g.

OUTPUT
1 (type of equipment) (*m*) BLOCKS
2 (type of equipment) (*m*) BLOCKS

The type of equipment may be

LINE PRINTER
TELETYPE
CARDS
FIVE-HOLE TELETYPE
ANY

where "Teletype" means a 7-hole (Teletype) paper tape punch,
"cards" mean a card punch
"any" means output on a line printer, Teletype punch, or cards.

The operators can control which equipments are used most by disengaging the other output equipments. *m* defines the limit of the output, and if the output exceeds *m* blocks of 4,096 characters, the program is stopped. If the number of blocks of output is not specified, it is taken as "1 block." Further, if there is only one output used, the output section may be omitted, and this is taken as if

OUTPUT
0 ANY 1 BLOCK

were included in the job description.

When printed, the output information itself is preceded by

OUTPUT (*n*)
(the title of the job)

and output of system information is always on output 0.

Downloaded from https://academic.oup.com/comjnl/article/4/3/226/380540 by guest on 13 March 2024

Job Description—Tapes

If a programmer uses magnetic tapes directly in his program (by use of tape instructions as distinct from using tapes in connection with input or output) then he specifies each tape used by two lines in the job description

TAPE

(*n*) (the title which is stored on block 0 of the tape)

where *n* is the programmer's number of the tape. When a new tape is required, the appropriate two lines of the job heading are

TAPE FREE

(*n*) (the title on block 0)

In this case, the title specified is written on Block 0 by the system.

If a file extends over several tapes, this is specified by a modified "tape" heading

TAPE/(*m*)

(*n*) (the title on block 0)

where *m* is the number of the continuation, counting from 1 upwards. The programmer's number *n* is the same for all *m*. The final tape of this file has the heading

TAPE/(*m*) END

If a program involves extensive input, then the job is preceded by copying this input to a magnetic tape. To initiate this copying process the input is headed

COPY TAPE FREE
(the title on block 0)

where the title specified is written on block 0. If a previously used tape is employed, the heading is

COPY TAPE (*b*)
(the title on block 0)

where *b* is the number of the tape block. (The programmer must always begin at the beginning of a tape block.)

Information may be read from this tape subsequently by specification of the tape and title of the information in the "input" section of the job description.

If a program involves extensive output then the output can be written on a private magnetic tape. This is specified in the "output" section of the job description as follows:

OUTPUT

(*n*) TAPE FREE/(type of equipment) (*m*) BLOCKS
(the title on block 0)

where *n*, "the type of equipment" and *m* are as for direct output, and where the title specified is written on block 0. If a previously used tape is employed, the specification is

(*n*) TAPE (*b*)/(type of equipment) (*m*) BLOCKS
(the title on block 0)

where *b* is the number of the tape block.

This private tape is printed by a steering tape consisting of

PRINT TAPE
(the title on block 0)

if the whole tape is to be printed, or

PRINT TAPE (*a*)/(*b*)/(*c*)
(the title of his output)

if one section of tape only is to be printed, from tape *a*, block *b*, word *c*.

Job Description—Miscellaneous

Further information may be given in the job description to indicate

1. the amount of core and drum store used,
2. the time for which the program is expected to compute,
3. the number of drums the program requires for programmed drum transfers.

All three apply to the execution stage of the program, i.e. excluding input from slow peripherals, compiling, and output to slow peripherals. These are specified by

	STORE	*s*	
	COMPUTING	*p.q*	HOURS
or	COMPUTING	*p.q*	MINUTES
or	COMPUTING	*p.q*	SECONDS
	DRUMS	*d*	

where *s* is the maximum number of core and drum 512-word blocks of store in use within the program during the execution stage, *p.q* is a fixed-point decimal number such as

COMPUTING 7·5 SECONDS

where the program is expected to run for not more than $7\frac{1}{2}$ seconds (if the estimate for store used and computing time is exceeded the program is stopped), and where *d* is the number of drums the program requires to reserve for programmed drum transfers.

If the total execution time is significantly different from the actual computing time, because there is considerable tape waiting time, the actual computing time should also be specified, e.g.

EXECUTION 5 MINUTES
COMPUTING 30 SECONDS

If information is not supplied in the job description, then

20 store blocks (10,240 words)
4 seconds computing time

and, of course, 0 drums are reserved. Estimates of the computing and execution times are taken as being equal unless both are specified explicitly.

Downloaded from https://academic.oup.com/comjnl/article/4/3/226/380540 by guest on 13 March 2024

End of Tape Markers

The end of a section of tape is indicated by

* * * (*x*)

where *x* is *Z*, *A*, *B*, *C* or *T*.

The marker

* * * *Z* indicates the genuine end of the tape/stack of cards

* * * *A* indicates "abandon previous incomplete section, if any" (this may be required by a machine operator)

* * * *B* indicates that a binary tape follows

* * * *C* indicates the end of a section, and that there is another section following on the same tape

* * * *T* indicates a temporary stop within a section.

The number of characters, *n*, on a binary tape may be indicated by

(*n*) * * * *B*

where *n* is a decimal number.

On reading the marker * * * *Z* the peripheral equipment is disengaged by the computer. When the operator next engages this equipment, a new section (with the appropriate heading and title) is read. The marker * * * *C* indicates the end of a section of tape, but the equipment is not disengaged and the next section is automatically read.

On reading the marker * * * *T* for a temporary stop, the equipment is disengaged as for * * * *Z*. However, when the operator next engages this equipment, a continuation of the current section (without a new heading) is read. Finally, on reading the marker * * * *B*, the computer reads the information following, in binary, without testing for further end-of-tape markers.

A better method of specifying the continuation of a section of data, without use of the marker * * * *T*, is by means of a modified "data" heading

DATA/(*n*)

where *n* is the number of the continuation of the section of data; e.g. for a program with data on two distinct paper tapes, the data may be headed

DATA/1
(the title of the data)

and

DATA/2 END
(the same title)

and each tape ends with the marker * * * *Z*. The continuation data tapes may be read into the computer in any order.

Acknowledgements

This work forms part of the Atlas project. It has benefited from many helpful discussions with the authors' colleagues at Manchester University and Ferranti Ltd., whose permission to publish is acknowledged.

Correspondence

To the Editor,
The Computer Journal.

Dear Sir,

"The LISP programming System"

From the paper by Woodward and Jenkins (1961) it appears that McCarthy's LISP system is aimed at bridging the gap between machine instructions and logical propositions in a way which on the machine side is dominated by questions of store utilization and access. It is widely agreed that any "generalized autocode" should have a recursive structure, and there are great advantages in a two-level system of direct language and meta-language. But as soon as one looks at the specific structure of LISP one notices that in the simplest presentation the hardware of the machine store must be such that two parts of each location are separately addressable. The subdivision of computer "words" into "syllables," which is becoming common in new machines, provides this facility; but it must be remembered that this adds to the complexity of the addressing system and so involves some additional cost. It is also essential that the "atoms" be small enough to pack two to a word, though in non-mathematical work the "atoms" may often be addresses of the storage of multi-word blocks of data (e.g. in sorting with detached keys) and two addresses per word is reasonable. Alternatively, the scheme could presumably be used with any machine code which has a Next Instruction Source address associated with each operand address, i.e. a 1 + 1 address code.

But if we are to be sufficiently mathematically minded to retain the idea of a *function*, I would suggest that it is useful to retain the distinction between "quantities" and "operators" in S-language. (I regard a function as a compound or "molecular" operator for this purpose.) Thus the authors' list X, +, COS, 7, (includes two quantities X and + and three operators, namely "add", "take the cosine of" and the opening bracket which is an organizational operator. In algebra a pair of brackets can be described as an operator meaning "treat the list between the mating pair of brackets as a single molecule." This is still true in the comma-and-bracket notation for LISP but cannot reasonably be applied to the dot notation, when a list (C, D, E) which would naturally be regarded as a molecule is represented by (C.(D.(E.Nil))). Looking at the machine format, it is possible to regard opening brackets as the equivalent of

(*Continued on p. 241*)

Downloaded from https://academic.oup.com/comjnl/article/4/3/226/380540 by guest on 13 March 2024