The output of AMCAP consists of binary machine code lines interspersed with modification lines. The modification lines are binary encoded and represent a compromise between the expense of storing and handling a large code and the expense of interpretively performing the modifications.

Parts of the ACSI-MATIC Programming System including parts of AMCAP have been coded for the MOBIDIC-A and Sylvania 9400 computers; some of the code has been debugged on the MOBIDIC-A. It is expected that an initial system will be operating by the end of summer.

Acknowledgment is given to the members of the Programming Research Group of the Astro-Electronics Division of Radio Corporation of America for their support and in particular to Roy Heistand for his contributions to the design and implementation of AMCAP.

REFERENCES

1. GURK, H., MINKER, J. The design and simulation of an information processing system. *J. ACM 8*, 2 (Apr. 1961), 260.
2. MILLER, L., MINKER, J., REED, W. G., SHINDLE, W. E. A multi-level file structure for information processing. Proc. Western Joint Comput. Conf., Apr. 1960.

Dynamic Storage Allocation in the Atlas Computer, Including an Automatic Use of a Backing Store*

John Fotheringham
Ferranti Electric, Inc., Plainview, New York

1. Introduction

This paper is concerned with the method of address interpretation in the Atlas computer. The Atlas has been designed and built as a joint exercise between the Computer Departments of Manchester University and of Ferranti Ltd., and the ideas and concepts described in this paper were conceived and developed by the University Computer Department. The address interpretation achieves the effect of dynamic storage allocation throughout the main memory; it also provides for automatic transfers of information, as required, between the core store and a backing store on drums. The core and drum stores can be considered together as a single memory (of maximum size about one million 48-bit words), and addressed as being entirely immediate access: access is in fact immediate only to the core part of the memory, but internal exchanges between the cores and drums are performed to bring the required information to the core store for use.

2. Some Details of Atlas

The word length is 48 binary digits, and most of the functions in the function code operate on full-length words or half-words of 24 bits. The central processor has one floating-point accumulator holding 87 bits and 128 index registers of 24 bits; some of the index registers have special purposes and only 90 are available for program use.

An instruction occupies a whole word, and consists of function digits, two index register addresses, and one store address. The latter is 24 bits long and occupies the second half-word of the instruction. The least significant three bits of the address are used to identify one of the eight 6-bit characters within a word for certain special functions, and the leading digit of these three is also used for identifying the half-word operand for 24-bit functions such as index register operations. The remaining 21 bits address a word, giving a range of over two million words; this range is divided into halves of which the first half is the main memory (core and drum) and the second half consists of various types of memory known collectively as the executive store. The leading address digit, therefore, can be considered as distinguishing between main memory and the executive store.

An important section of the executive store is a fast read-only memory, the contents of which cannot be changed by program. Amongst other things, this memory holds a housekeeping program, known as the SUPERVISOR, which controls the operation of the whole system. All input and output, transfers between different types of storage, monitoring, bookkeeping and checking are performed by the Supervisor program.

The SUPERVISOR is called into action either by an object program, by means of one of a number of special functions, or by an interrupt signal generated by the system when some housekeeping action is required.

One instance of an interrupt signal being generated is when a reference to the main memory refers to a word which is on the drum.

3. Address Interpretation in Main Memory

The term *address*, when applied to the main memory of Atlas, has a slightly more precise meaning than usual: an *address* is an identifier of a required piece of information but not a description of where in main memory that piece of information is. The locating of the information is done by a comparison table search process applied to the address.

* Presented at an ACM Storage Allocation Symposium, Princeton, N. J., 23–24 June 1961.

The main memory registers are grouped in units of 512, but a careful distinction must be drawn between a 512-word block of information and a 512-word unit of memory; the latter is described as a page of core store or a sector of drum store. Thus a block of information occupies either a page of core store or a sector of drum store. The programmer need not think of addresses in this way, but it is how they are treated by the machine. The address of a word in main memory is 20 bits long (not counting the most significant address bit, which is zero for main memory), and it is effectively an 11-bit block address, followed by a 9-bit position address within the block.

Each 512-word page of core store has associated with it an 11-bit register, known as a *page address register*, which is used to hold the number of the block which is in that page. An address is decoded by comparing the 11-bit block address with the contents of each of the page address registers in parallel to locate the block, and then the 9-bit word position part is decoded in the normal way.

The page address registers are set by the SUPERVISOR. The SUPERVISOR controls the loading of the core store: when it needs a block in the core store, it brings the block to any available page of cores and sets the corresponding page address register to the number of that block. A program can use any addresses in the 20-bit range that are convenient, so long as the SUPERVISOR is notified, and when the SUPERVISOR brings information to the core store it arranges for that area of core store to have the appropriate "addresses". For information on the drums, the SUPERVISOR keeps a table of where each block of information is, so that when a block needs to be used, the SUPERVISOR can transfer it to an available page of core memory for use. The problem is, of course, to find an available page.

4. Drum Transfer Learning Program

If the comparison of block number with the contents of the page address registers fails to produce an identity, the SUPERVISOR is called in by an interrupt signal, and must bring the required block into the core store. To make room for this block, the SUPERVISOR will probably also have to write away a block of information to the drum, and the DRUM TRANSFER LEARNING program is the section of the SUPERVISOR which estimates which block can best be spared from the cores.

The learning program is described in detail in a paper entitled "One Level Storage System" by T. Kilburn, D. B. G. Edwards, M. J. Lanigan and F. H. Sumner.[1]

5. Dynamic Storage Allocation

In addition to allowing the whole memory to be treated as a unified store, with each word addressible, the addressing system gives several advantages in the use of this store, viz:

(1) A programmer, or a compiler, can use any addresses in the 20-bit range that are convenient, and can divide his storage into several areas with widely spaced addresses. This means he can conveniently allow for the maximum possible size of each area, and yet on any particular run his program needs only just as much store as it actually is going to use.

(2) A program written for one machine will run, without alteration, on a second machine with a different division of main memory between core and drum store, so long as there is adequate memory overall. Similarly, if additional core store is attached to a machine, all programs will automatically take advantage of the additional fast memory without any modification.

[1] Submitted to a technical journal for publication.

Experience in Automatic Storage Allocation*†

George O. Collins, Jr.

Computer Associates Inc., Woburn, Massachusetts

Introduction

The purpose of this paper is more to present a point of view than to give details of storage allocation techniques. This point of view is that in order to efficiently solve some of the extremely complex problems being considered today, one must have a powerful programming system not only to assist in the preparation of program and data segments but also to supervise and participate in the execution of the programs. To achieve the power and flexibility needed, the programming system must provide a facility

* This work was supported in part by a subcontract from Technical Operations, Inc., under a contract with the Department of the Air Force, AF33(600)-41896.

† Presented at an ACM Storage Allocation Symposium, Princeton, N. J., 23–24 June 1961.