Outline of the Logical Design of the ZAM-41 Computer*

L. LUKASZEWICZ†

Summary—The ZAM-41 is a medium-sized, high-speed, parallel, binary computer with the word length of 24 bits and the internal storage capacity from 8192 to 32,768 words. It is destined mainly for data processing, but it may also be used for scientific computations and for real-time control. The full effect of multiprogramming has been obtained on the above computer by means of a small amount of hardware and an appropriate supervisor program. The problem of mutual interference of concurrently performed programs has been particularly taken into account.

Introduction

THE PURPOSE of designing the ZAM-41 computer was to produce an economical machine, its scope of application being as great as possible. The ZAM-41 was assumed to serve, depending upon its configuration, as a small or middle-sized computer for: 1) data processing, 2) scientific computations, or 3) controlling objects in real time. The effectiveness of the computer for each of the above-mentioned applications should be reasonable, and the amount of dispensable hardware reduced to the minimum.

According to the analysis performed, the problem has been solved successfully by means of developing a simple high-speed binary computer with a relatively short word, on which more complex operations may be defined by using subroutines. Great effectiveness of subroutines defining data processing operations could be attained by means of an appropriate list of the computer instructions. A more significant decrease of the computer speed appears in scientific computations, this results from subroutines being used for floating point operations. However, this decrease is recompensed to a certain degree by the simple and economical structure of the computer. The computer speed is especially favorable when controlling objects in a real time.

Most effective use of relatively slow input and output devices is needed on a data processing computer. The minimal requirement is the possibility of a concurrent reading, writing and computing within one program. The enlarged requirement is the possibility of executing several of these programs concurrently (multiprogramming). Mutual coordination of the structure of the supervisor program and the computer organization in the stage of designing is indispensable in order to solve the above problem by means of a small amount of hardware.

* Received August 30, 1963.
† Institute of Mathematical Machines, Polish Academy of Sciences, Warsaw, Poland.

The present paper gives an outline of the ZAM-41 organization. The ZAM-41 is the successor of ZAM-2 that is being produced in Poland. The ZAM-2 is a small tube computer destined for scientific and technical computations. Many of the ZAM-41 subroutines, especially those for the supervisor program, have been checked by means of a simulation performed on the ZAM-2 computer.

General Computer Configuration

The typical set of ZAM-41 units applied for data processing is shown in Fig. 1. In the central part of the computer the 24 bits of information, plus 1 parity bit, are exchanged parallelly through a common bus joining independent computer units. Several local actions may be executed on the computer concurrently; for instance, the transfer of information through the bus from synchronizer to ferrite store and the multiplication performed in an arithmetic unit.

The set of ZAM-41 external devices might be smaller, consisting, for instance, of one block of ferrite storage, one paper tape reader, one punch unit and one magnetic drum connected with the computer by a selector.

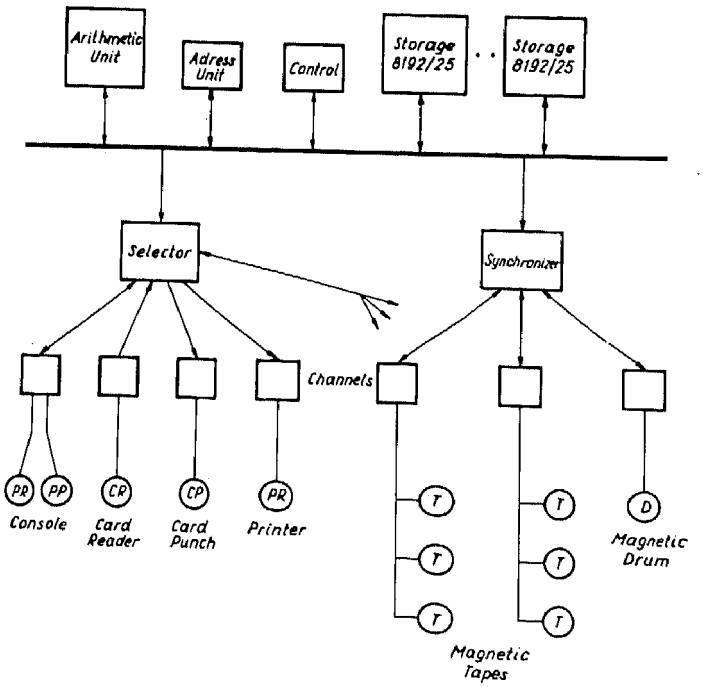

Fig. 1—Typical ZAM-41 configuration for data processing.

The Information Formats

The basic information unit of ZAM-41 is a 24-bit word that may represent:

a) The computer instruction, its structure shown in Fig. 2. The significance of separate parts of the instruction is the following:
   A—15-bit *address* part that permits addressing up to 32,768 words;
   R—6-bit *operational* part making possible to distinguish 64 various instructions;
   M—*modification* bit indicating the addition of the B register content to the address part of the instruction;
   P—the bit indicating *indirect addressing;*
   D—the bit indicating *modification* of the address part by the addition of the content of Lower Boundary register (*LB*). This modification occurs only when digit 1 is written in the Legality Indicator (*L*). Further details on this subject will be given.
b) Four 6-bit *characters*, for instance, ABCD.
c) A *number* consisting of a sign and of an absolute value expressed by 23 bits.

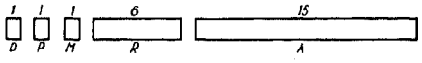

Fig. 2—ZAM-41 instruction format.

The ZAM-41 words written in two neighboring computer locations may represent a 48-bit floating point number. The exponential part of this number consists of 9 bits, its fractional part of 39 bits, which correspond to more than 11 decimal digits. Operations on floating point numbers are executed by means of appropriate subroutines. A group of special instructions is provided to render these operations efficient. For instance, an addition of two floating point numbers involves only ten computer instructions. Operations performed on floating point numbers, composed of four basic computer words, may also be executed efficiently. Then, the exponent would consist of 9 bits and the fractional part of 70 bits, which corresponds to more than 20 decimal digits. The time of such operations is approximately twice as long as in the case of numbers represented by two computer words.

Automatic Program Interruption

The program interruption system in ZAM-41 is the result of a concurrent use of several input and output devices and of multiprogramming. The list of numbered conditions causing the program interruption is the following:

0 Central processes error or disturbances in the main power supply.
1 Illegal instruction.
2 Error in one of the input or output devices.
3–31 Request of attention towards input or output devices.

Program interruption caused by 3–31 may occur only if an appropriate indicator of interruption is switched on. If several conditions are signaled simultaneously and the indicator of the interruption is on, the condition chosen is that of the lowest interruption number. Conditions 0 and 1 always cause interruption, independently of the state of the indicator of interruption.

The automatic program interruption system initiates the following actions:

1) Completing the running instruction;
2) Storing the content of the instruction counter augmented by 1 in the storage location indicated by the content of the *LB* register;
3) Transfer to the instruction recorded at the address $128+n$ (where n is one of the above given condition numbers). Usually, this transfer causes a further transfer to one of the supervisor subroutines.

Instructions of the Central Processing Unit

The accumulator *A* and the multiplier register *M* are basic 24-bit arithmetic registers. They are often treated as one common double-length register. Beside the most frequent arithmetical and logical instructions performed in those registers, special instructions enabling such operations as storage search or 6-bit character conversion appear in ZAM-41. A group of address and control registers is shown in Fig. 3. The significance of the instruction counter *IC* and *B* register is the same as in the majority of computers. Besides, there is the following group of basic registers and indicators connected with computer multiprogramming:

Lower Boundary register (*LB*),
Upper Boundary register (*UB*),
Legality indicator (*L*).

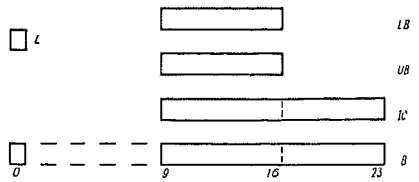

Fig. 3—Group of control and address registers.

Every *LB* and *UB* register comprises only eight more significant positions, the remaining seven less significant positions are assumed to comprise only zeros. The register contents *C*(*LB*) and *C*(*UB*) define the storage area inside of which the currently running program acts. The values *C*(*LB*) and *C*(*UB*) are determined by the supervisor in the moment of transferring to the corresponding program. The supervisor always occupies the storage area with lowest addresses. A typical situation of program distribution in the storage is shown in Fig. 4.

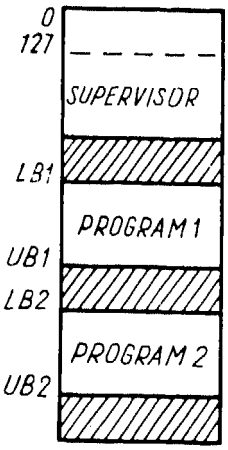

Fig. 4—Typical program distribution in the storage.

The indicator L is related to the notion of legality and illegality of the instruction. If $L=1$ all instructions are legal, if $L=0$ some instructions are illegal and cause a program interruption. The state $L=1$ is usually connected with the action of the supervisor. The state $L=0$ is connected with the performance of programs.

Instructions are always illegal if they can interfere with other programs, thus,

1) Instructions with effective addresses exceeding the area, defined by $C(LB)$ and $C(UB)$;
2) Instruction STOP and instruction causing the change of LB and UB contents;
3) All instructions connected with input and output.

The only exception from the above rule is the instruction LEGALIZE AND JUMP TO n which is legal if and only if $n<128$. In case of legality this instruction causes $L=1$ and the transfer to one of the 128 storage locations indicated by n, where a further transfer instruction to one of the supervisor subroutines is comprised as a rule.

It follows from the above that in consequence of performing an arbitrary instruction of a running program, one may only move in the area allowed for the program or transfer to one of the supervisor subroutines. The latter case may occur on purpose as a result of the instruction LEGALIZE AND JUMP, or it may result from an illegal instruction that has been used.

Supervisor subroutines are standard computer equipment and their arrangement fully ensures against mutual program interference resulting from the use of an erroneous instruction in any of these programs.

If $L=1$, and $D=1$ in the performed instruction, then D modification acts by adding the content of LB register to the instruction address part. Moreover, in case of the JUMP instruction, zero is written on the indicator L, this instruction is used to cause the transfer from the supervisor to one of the programs. Using the D modification every supervisor subroutine may easily call for working locations and parameters in the storage area assigned to the executed running program. Moreover, the same supervisor subroutine may be easily used by several programs simultaneously.

The D modification does not act during the performance of the running program since then $L=0$ as a rule. The D modification bit may be used in these programs for other purposes, for instance, to indicate instructions with addresses which should be transformed when loading the program into the ferrite storage.

Input and Output System

External computer devices are grouped into channels, the maximum number of which may be sixteen. Every channel may contain up to eight so-called external registers which fulfill their functions as 1) control registers for the given channel, 2) buffer registers for information transferring, 3) indicator registers showing the actual state of the given channel.

The information from an arbitrary storage location is transferred to an arbitrary external register via selector by means of two instructions. The first of them puts the nth storage location content into the intermediary W register which is in the selector. The second one puts the content of the W register into a chosen external register. Likewise, using two similar instructions, the content of an arbitrary external register is transferred to an arbitrary storage location.

Using the above instructions and possibilities provided by program interruption, the transferring of the block from the slow input device which is connected to the selector may be simulated. Let us consider, for example, the reading of cards column by column. Assume that 80 characters recorded on one card represent a block of information of 20 words, each of them comprising 4 characters. Card reading starts by introducing control information into the external control register in the card reader channel. Every consecutive four characters that are read fill out the external buffer register connected with the card reader. This causes an automatic program interruption and a transfer to the supervisor subroutine connected with the card reader. The supervisor takes the word, recorded in the buffer register, and puts it into the consecutively computed location of the storage field destined for the card that is being read. Then, the supervisor causes the return to the program that was interrupted by the card reader. This action is repeated until the whole card is read, and then, the transfer occurs to the central supervisor subroutine. This subroutine causes the transfer to the program which is chosen depending upon priorities assigned to all programs and their actual state.

The above simulation of transferring of the block from a slow input or towards a slow output device brings about only a slight loss of the central unit time. On the other hand, hardware connected with those device channels is significantly reduced.

In case of a high-speed input or output device, as for

instance, a magnetic tape, the transferring of the block in ZAM-41 is automatic. For example, let us consider the operation of the recording of the block of information on a magnetic tape. Let us assume that initially this block was recorded in the form of several segments distributed in separate locations of the ferrite storage. The above operation is initiated by the loading of the external control register of the channel with information which starts the action of recording on a tape. The initial address of the word chain, controlling the transfer of consecutive segments, is simultaneously loaded into the second external register of the tape channel. The initial address and the word count of the segments are comprised in control words (Fig. 5). The tape channel takes consecutive control words and causes automatic sequential recording of segments in one block of the tape. The operation ends at the moment when the control word arrived at contains zero in the word count position. The very moment when the writing of the block is terminated, the running program is interrupted and the transfer to another program becomes possible.

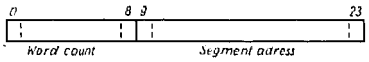

Fig. 5—Control word format.

As is seen from the above, one running program may initiate the transfer of the block of data which should continue in case of a transfer to another program. Therefore, the current content *LB* and *UB* cannot check the transfer of data from the input device to the ferrite storage and vice versa. Thus, to avoid the introducing of data by one running program into the area reserved for another program, it has been accepted that input and output operations may take place only through supervisor subroutines. The assignment of external devices and of storage locations into which data are introduced to the programs occurs at the moment of loading the programs into the ferrite storage by the operational system. Supervisor input and output subroutines may be called in by progams only by means of the LEGALIZE AND JUMP instructions.

Conclusion

By means of an extensive use of appropriate subroutines on the high-speed simple binary ZAM-41 computer, many effects are arrived at that appear as a rule on computers of a more complex structure. The field of ZAM-41 application is very large, the effectiveness being reasonable in every case, especially for data processing. Subroutines of floating point operations permit to operate on numbers of a normal or increased precision. Full effects of multiprogramming have been obtained as the result of the full use of possibilities provided by the program interruption. Thanks to the appropriate structure of the supervisor program the amount of hardware connected with multiprogramming was reduced to a minimum. Simulation of the transferring of the block enables a relatively simple channel solution of slow input and output devices.

Acknowledgment

The design of the ZAM-41 computer was elaborated by a large group of scientific workers of the Institute of Mathematical Machines of the Polish Academy of Sciences, Warsaw. The author wishes to thank Dr. St. Majerski, and Messrs. Głowacki, Englert, and Łabanowski of the Organization Department, for their contributions to the computer organization, as well as J. Swianiewicz, E. Zaborowska, and R. Rędziejowski of the Department of Programming, for their contributions to the supervisor program.

Reference

[1] W. Buchholz, Ed., "Planning a Computer System," McGraw-Hill Book Co., Inc., New York, N. Y.; 1962.