# THE LOGICAL STRUCTURE OF THE RC 4000 COMPUTER

PER BRINCH HANSEN

(1967)

**This paper describes the logical structure of the RC 4000, a 24-bit, binary computer designed for multiprogramming operation. The design is characterized by multiple accumulators, expandable storage, low-speed and high-speed data channels, storage protection, and program interruption. Processing operations include byte manipulation, word comparison, and integer and floating-point arithmetic. Typical operation times are from 2.5 to 5.5 microseconds.**

## Introduction

The RC 4000 is a medium-sized computer, designed by Regnecentralen for real-time control, numerical computation, and administrative data processing. The RC 4000 was developed in 1966–67 as part of a process control system for the Danish engineering company Haldor Topsøe (Brinch Hansen 1967).

Although the RC 4000 was influenced by a number of existing computers (Buchholz 1962, Amdahl 1964, Control Data 1965), its design has merits of its own, in particular the homogeneous register structure and the clean, systematic instruction set.

This paper describes the logical structure of the RC 4000, explaining the choice of data and instruction formats, storage addressing, arithmetic, input/output control, storage protection, and program interruption.

P. Brinch Hansen, The logical structure of the RC 4000 computer, *BIT 7*, 3 (1967), 191–199. 

### Design objectives

It was felt desirable to design the RC 4000 for a broad range of applications. Consequently, several data formats, ranging from 12 to 48 bits, are available for integer and floating-point arithmetic. Input/output data channels are available for transmission of single words from low-speed devices and transfer of blocks from high-speed devices.

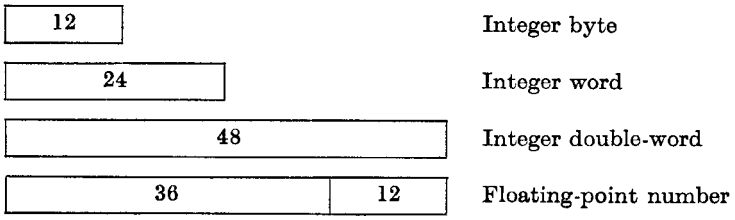
An important objective was to allow for future expansion of the system. This implies the ability to address a large internal store directly as well as the standardization of input/output control to permit the connection of new kinds of peripheral devices.

In real-time applications, multiprogramming is assumed to be the normal mode of operation. In the RC 4000, multiprogramming is assisted by concurrency of program execution and input/output operations, program interruption and program protection.

In choosing the instruction set, a primary goal was to improve the efficiency of symbol manipulation and code generation performed by compilers. The former task is facilitated by byte manipulation and word comparison, the latter by the use of a single, simple instruction format.

### Data formats

Four data formats are available for byte manipulation, integer arithmetic, and floating-point arithmetic:




Bytes of 12 bits are directly addressable. This ensures efficient storage of analog input data in process control applications. Byte addressing is also a powerful tool in the manipulation of symbols encountered in file maintenance activities and program translation.

The basic operand in integer arithmetic is a 24-bit word. This word length is sufficient for all address computations and for most integer arithmetic in process control applications.

Double-length integers of 48 bits and floating-point numbers with 36-bit fractions and 12-bit exponents are provided for administrative data processing and numerical computation.

**Register structure**

In the RC 4000, the attempt to create a homogeneous register structure goes considerably beyond those found in earlier computers. The RC 4000 has four working registers of 24 bits each; three of these also function as index registers. Removing the distinction between accumulators and index registers makes the full instruction set available for immediate address computation, while the availability of several accumulators drastically reduces the need to store intermediate results.

To utilize multiple accumulators effectively, it is necessary that operations can be performed directly between two registers. This has been achieved by making the registers addressable as the first four words of the internal store. There are no restrictions on the use of registers as storage operands; it is even possible to execute instructions in them. To make the analogy between registers and storage words complete, each register is supplied with a protection bit.

In double-length operations, two adjacent working registers contain an operand of 48 bits. The registers are connected cyclically as double-length registers.

**Address modification**

The efficiency of programming is closely connected with the possibility of address modification within instructions. In the RC 4000, address modification includes indexing, relative addressing, and indirect addressing.

A common data processing problem is the analysis and transformation of text strings (as in program translation). In a computer with an addressable store, the most general way of implementing a data transformation is to use table look-up. The key to table look-up is the ability to modify addresses by the value of data being processed. This requirement is met efficiently through the use of index registers.

The ability to relocate programs in the internal store is necessary in applications where the library of programs is kept on a backing store and only brought to the internal store when active. Dynamic relocation requires that the addressing within a program be made independent of its current

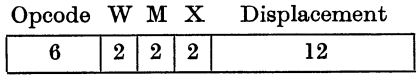
location in the store. This is achieved by relative addressing, implying a modification of the address part by the contents of the instruction counter.

Communication between relocatable programs and data tables is facilitated by indirect addressing. When new segments are loaded, their base addresses can be placed in a table through which references are made by means of indirect addressing.

**Instruction format**

An important objective in the design of the RC 4000 was to simplify the tasks of programming, compilation, and debugging. This calls for a uniform and systematic instruction set that prevents trick-coding. Consequently, we have restricted the modifications of instructions to include only the selection of an accumulator and the specification of address modification. The possibility of having several instruction formats was rejected on the grounds that it would complicate the compilation of programs. For the same reason, we have resisted the temptation to expand the instruction set with ad hoc solutions such as using a part of the effective address to specify whether a shift operation is arithmetical or logical.

The RC 4000 instruction format is divided into an operation byte and an address byte, each of 12 bits:




The operation byte specifies 64 basic operations, a result register $W$, an addressing mode $M$, and an index register $X$. (An index designator equal to zero means no indexing.)

The address byte specifies a displacement from $-2048$ to 2047 bytes within the program. This is adequate for the majority of addresses, but insufficient for the direct addressing of the entire store. An effective address $A$ of 24 bits is generated as follows:

| | | |
|---|---|---|
| $M = 00$ | $A = X + D$ | direct addressing |
| $M = 01$ | $A = word[X + D]$ | indirect addressing |
| $M = 10$ | $A = X + IC + D$ | relative addressing |
| $M = 11$ | $A = word[X + IC + D]$ | relative-indirect addressing |

In the address calculation, the displacement $D$ is treated as a 12-bit

signed integer that is extended toward the left to 24 bits, before being added to the index register $X$ and the instruction counter $IC$.

In storage access operations, the effective address is treated as an unsigned integer of 24 bits. Thus, the upper limit to the expansion of the store is 16,777,216 bytes.

**Arithmetic**

The following is confined to integer arithmetic. (Floating-point arithmetic has not yet been implemented.)

Integer operands are represented in the two's complement notation. This number representation is particularly suited to arithmetic with mixed byte and word operands; in byte arithmetic, 12-bit integers are extended to 24 bits simply by a duplication of the sign bit toward the left.

All arithmetic operations except multiplication indicate an overflow of the result. Addition and subtraction also indicate a carry from the sign position. The carry and overflow indications are given in a 2-bit register, called the exception register. The exception bits can be tested individually by the instruction SKIP IF NO EXCEPTIONS. The carry indication simplifies the programming of multiple-length arithmetic. This is further facilitated by the generation of a 48-bit product in multiplication, the use of a 48-bit dividend in division, and double-length arithmetic shifting.

**Input/output control**

Character-oriented devices such as typewriters, paper tape readers, and punches are connected to a single, low-speed data channel, which communicates directly with the internal working registers. Each device has a buffer register of 24 bits. Transfers of data words between working registers and device buffers take place one at a time under program control. Transfers between buffer registers and external data media are, however, controlled independently by the devices; thus several such transfers can take place simultaneously.

Input/output operations are initiated by a single-instruction, the effective address of which specifies a device number and a device command. Program execution continues immediately after an input/output instruction, with the exception register set to indicate whether the operation was initiated or rejected. An input/output operation is rejected if the device in question is non-existent or busy with a previous operation.

There are four basic input/output commands: read, write, control, and sense. A read command initiates the input of a character to a buffer register. The write and control commands transfer the contents of a working register to a buffer register and initiate output and control operations, respectively.

A sense command requests a device to transfer a status word from its buffer to a working register. The exception register indicates whether the status word is available; this is the case only after the termination of an operation. The status word contains the last input or output character as well as information about parity errors and other exceptional events.

In 1967, the RC 4000 will be extended with a high-speed data channel for handling a magnetic drum, tape stations, card readers, and line printers. This channel provides multiple block transfers directly to or from the internal store concurrent with program execution. A multiplexer switches rapidly among the devices, connecting them to the high-speed channel whenever they are ready to transfer a data word.

The high-speed devices will use the low-speed channel to transfer commands and addresses of buffer areas in the store. All peripheral devices are thus controlled by the standard input/output instruction, and the high-speed devices are simply addressed in continuation of the low-speed devices.

**Program interruption**

In a multiprogramming system, programming should be just as straightforward as in purely sequential programming. This requires that time-sharing among independent programs be handIed by a monitor program that has complete control of the system. To ensure regular and automatic retum to the monitor program, a program interruption system is needed in connection with a real-time clock. Program interruption is also desirable for monitoring of such infrequent events as violation of the storage protection and arithmetic overflow.

In the RC 4000, interrupt sources are connected to an interrupt register of 24 bits. A mask register permits individual enabling and disabling of the interrupts. The interrupts are examined after each instruction cycle. An interruption is performed as follows: the interrupt bit is turned off, and its position within the register is stored as an integer in a fixed location; the instruction counter is then stored in another location, and an indirect jump is made to a fixed address. In the case of simultaneous interrupts, the left-most signal is served first.

The entire interruption system can be disabled for short intervals, when

an interruption would be awkward (e.g. while a previous interrupt is being processed). When the system is disabled, interrupts are still registered, but not served. The interrupts are automatically disabled when the monitor program is entered. They can be enabled again (or disabled) by the instruction JUMP WITH INTERRUPT ENABLED (DISABLED).

The interrupts can be classified according to priority as follows:

| | |
|---|---|
| First: | Protection Violation |
| Second: | Integer Overflow |
| Third: | Floating-Point Overflow |
| Fourth: | External Interrupts |

Protection violations are recognized in the following situations: execution of an unassigned operation code; reference to a non-existent storage location; attempts to violate program protection.

The right-most 21 bits in the interrupt register can be assigned to external signals from real-time clocks, operator keys, and input/output devices.

**Program protection**

Monitor control of the RC 4000 is guaranteed by storage protection and privileged instructions.

Storage protection is achieved by means of an additional bit in each storage word. The monitor program consists of all storage words in which the projection bits are set. An unprotected program can neither alter nor jump to a protected word. Attempts to violate storage protection cause program interruption. Thus the monitor can neither be destroyed by subordinate programs, nor entered at arbitrary points.

Further protection is achieved by privileged instructions that can only be executed within the monitor. These instructions include input/output, setting of protection bits as well as enabling and disabling of interrupts.

The protection system can be summarized as follows: a program interruption sets the computer in the monitor mode and starts the monitor program at a well-defined point. In the monitor mode, all instructions can be executed as long as they are protected. The computer returns to the task mode, when the first unprotected instruction is executed. In the task mode, program interruption results if the following is attempted: storing to a protected location; jumping to a protected location (by explicit branching or by sequential program execution); execution of a privileged instruction.

**Instruction set**

The instructions available in the RC 4000 are listed in the appendix. The most interesting are explained in the sequel.

To reduce the frequency of index loading, the instruction MODIFY NEXT ADDRESS was introduced. This instruction modifies the displacement in the following instruction by its own effective address. If indirect addressing is used in the modify instruction, the effect is equivalent to indexing by the contents of a storage location. A series of modify instructions can add the values of several working registers for indexing; it can also serve as multi-level indirect addressing.

Index registers are conveniently initialized and incremented by the instruction LOAD ADDRESS. LOAD ADDRESS COMPLEMENTED permits incrementing and sign reversal of a register. Bytes can be loaded with zero extension, sign extension, or no extension. Useful too, is an instruction that exchanges the contents of a register and a storage word.

Standard instructions are available for arithmetic, Boolean operations, and shifting. In double-length logical shifts, the cyclical connection of working registers greatly simplifies the packing of series of bit-fields within a word.

A single instruction, JUMP WITH REGISTER LINK, combines the functions of direct jumping and procedure calls: it places a return address in a working register, provided the register designator is not zero. This form of procedure call has two advantages: a procedure can be entered at several points, and parameters can be transferred by indexing when needed.

Conditional transfers are based on a systematic set of compare and skip instructions, which leave the working registers unchanged. The skipping of the following instruction can be conditioned on the arithmetic relation between the effective address and the value of the working register (e.g. greater, less, equal, or unequal). The effective address can also be used as a mask to test selected register bits for the conditions: all zeros, and all ones.

Initial program loading is started by an operator key, which causes the input of four 6-bit characters to working register 0, followed by a jump to the register. Boot-strapping is continued by the autoload instruction, which reads four characters into an addressed storage word.

## Appendix: Instruction set

*Address handling*
- Modify Next Address
- Load Address
- Load Address Complemented

*Register Transfer*
- Load Half Register
- Store Half Register
- Load Register
- Store Register
- Load Double Register
- Store Double Register
- Exchange Register and Store

*Integer Arithmetic*
- Load Byte with Zeros
- Load Integer Byte
- Add Integer Byte
- Subtract Integer Byte
- Add Integer Word
- Subtract Integer Word
- Multiply Integer Word
- Divide Integer Word
- Add Integer Double-Word
- Subtract Integer Double-Word

*Floating-point Arithmetic*
- Convert Integer to Floating
- Convert Floating to Integer
- Add Floating
- Subtract Floating
- Multiply Floating
- Divide Floating

*Logical Operations*
- Logical And
- Logical Or
- Logical Exclusive Or
- Shift Single Arithmetically
- Shift Double Arithmetically
- Shift Single Logically
- Shift Double Logically
- Normalize Single
- Normalize Double

*Sequencing*
- Jump with Register Link
- Skip if Register High
- Skip if Register Low
- Skip if Register Equal
- Skip if Register Not Equal
- Skip if Register Bits One
- Skip if Register Bits Zero
- Skip if No Exceptions
- Skip if No Protection

*Monitor Control*
- *Jump with Interrupt Enabled
- *Jump with Interrupt Disabled
- *Clear Interrupt Bits
- Store Interrupt Register
- *Load Mask Register
- Store Mask Register
- Load Exception Register
- Store Exception Register
- *Clear Protection Bit
- *Set Protection Bit
- *Input/Output
- *Autoload Word

*denotes privileged instruction

## Acknowledgments

The logical structure of the RC 4000 was designed by Peter Kraft and the author. The hardware implementation was supervised by Henning Isaksson. Our close cooperation throughout the project with members of Mr. Isaksson's group, especially Allan Giese and Mogens Strange, proved invaluable.

We are also indebted to Christian Gran, Jørn Jensen, Peter Naur, Bjarner Svejgaard, and Povl Villumsen for valuable comments on the design.

## References

Brinch Hansen, P. 1967. The RC 4000 real-time control system at Pulawy. *BIT 7*, 4, 279–288.

Buchholz, W. (ed.) 1962. *Planning a Computer System: Project Stretch*, McGraw-Hill, New York.

Amdahl, G. M. 1964. The structure of System 360, Part III—Processing unit design consideration. *IBM Systems Journal 3*, 2–3.

*Control Data 1700 Computer System* 1965. Computer Reference Manual, Control Data Corporation, (September).