

TR 00.1318  
August 30, 1965

# Technical Report

The Design of Processor Controls Using  
a Read-Only Storage

Dorothy L. Schnabel

## The Design of Processor Controls Using a Read-Only Storage

by

Dorothy L. Schnabel

### ABSTRACT

This paper describes some of the considerations made in the design of the control of a processor using a read-only storage.

Using an IBM System/360 Model 50 as an example, the paper shows how a read-only storage (ROS) word is used to control the processor action in one state and provide sequence information for determining the next state.

The Model 50 ROS word is divided into four segments, each having the following chief function.

- Segment one--supplies the data flow control information for action in the present state.
- Segment two--supplies control information for remembering significant states and state events which will determine future action.
- Segment three-- supplies a constant for use as control or data information.
- Segment four--provides information for determining the address of the next control word to be selected from the ROS.

Emphasis is placed on the design of the control word structure and its decoding rather than on microprogramming techniques used in the execution of program instructions.

IBM 0360-50  
Storage  
Microprogramming  
Read-Only  
07 Computers



International Business Machines Corporation  
Systems Development Division, Poughkeepsie, New York

## CONTENTS

|                                               |    |
|-----------------------------------------------|----|
| IBM System/360 Model 50 CPU                   | 1  |
| The Design of the Model 50 Processor Controls | 2  |
| Read-Only Storage Word                        | 3  |
| Specification of Data Flow Controls           | 6  |
| Increasing the Bit Efficiency of the ROS word | 8  |
| Status Control Field                          | 8  |
| Emit Field                                    | 9  |
| Read-Only Storage Word Addressing             | 10 |
| Specification of Branch Test Micro-Orders     | 12 |
| Conclusions                                   | 14 |

# IBM SYSTEM/360 MODEL 50 CPU

As shown in Figure 1, there are two parallel data paths in the Model 50. One is a 32-bit wide binary adder and shifter path. The L register feeds into the left side of the adder (X) via a true or complement gate, (T/C) and the R, M, and H registers feed into the right side of the adder (Y). The L, R, M, and H registers are also 32 bits wide.

The adder output can be shifted either right or left by one or four bits. The shifter output latch (T) goes to a bus feeding the L, R, M, and H registers and also to the main storage address register, SAR, and the main storage data register, SDR.

The Model 50 has a maximum 256K byte main storage, also 32 bits wide, which operates on a 2.0- $\mu$ sec. cycle. The storage is addressed from a 24-bit storage address register. Data is transferred to and from storage via the storage data register.

A 24-bit instruction address register and counter feeds into SAR.

The other parallel data path is 8 bits wide and is called the mover. It performs logical operations such as the AND, OR, and EXCLUSIVE OR of 8-bit data groups

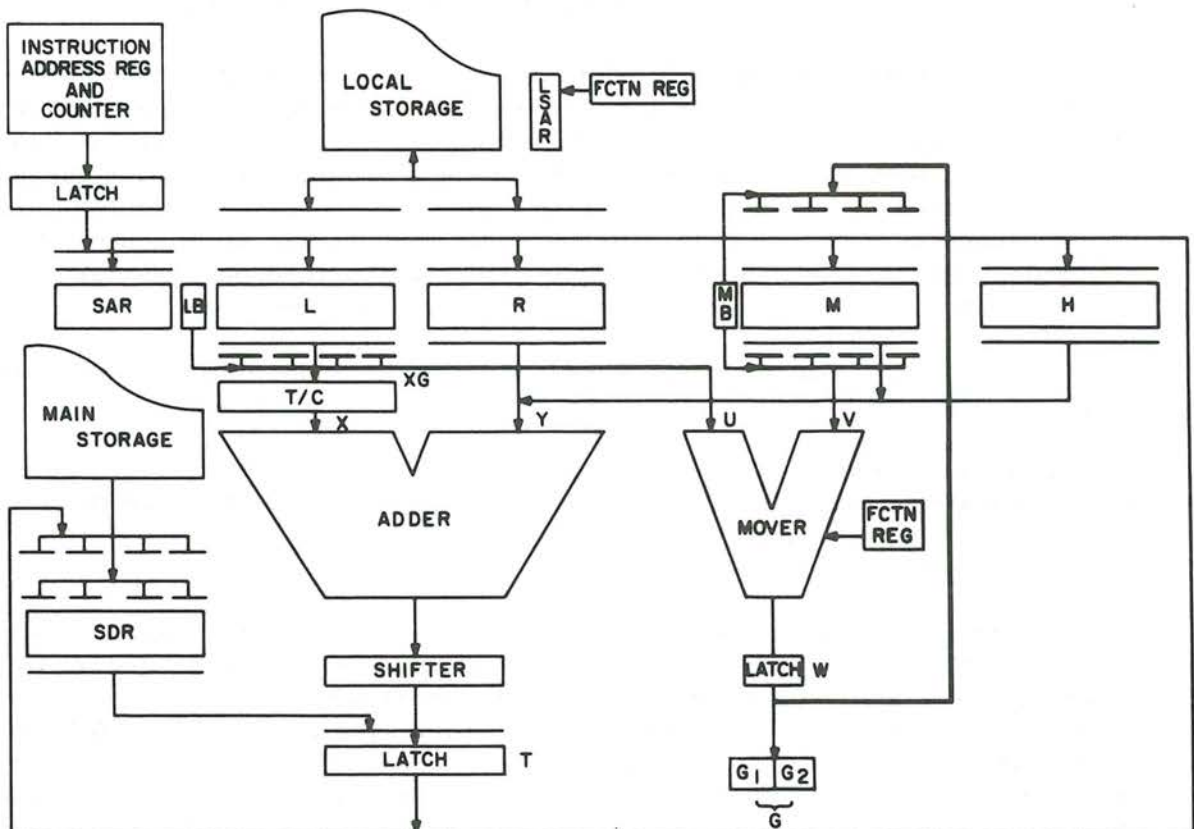


Figure 1. IBM System/360 Model 50 simplified data flow.

The controls in a ROS machine tend to be more formalized than those in a hardware machine. This proves to be a great advantage in the simulation of the control design before implementation.

The fact that the control specification is largely in a storage device makes design changes much easier. A storage unit plane can be replaced without any hardware changes.

Once the gating controls have been defined for a data flow, all microprogrammers are free to activate this gating. Let us assume that ten engineers are assigned to design a hardware-controlled machine and that each wishes to activate a particular gate, say an adder input, under 20 different conditions. A two-hundred way logical OR could be used to gather all conditions, or the ten designers could integrate their logic, making debugging, ease of understanding, and change very difficult.

This problem does not exist in a microprogram-controlled machine. A gate may be activated by each word in the ROS and conceivably can come from thousands of conditions.

It is the purpose of this paper to show some of the general considerations made in the design of a ROS-controlled machine and yet cite specific examples from the design of the System/360 Model 50 computer.

## READ-ONLY STORAGE WORD

The Model 50 ROS word has four control segments as shown in Figure 2.

The first segment contains encoded data which will control all data flow action such as the gating to and from registers, the insertion of a carry into an adder, counter control, and the selection of main and local storage.

The second segment will be called status control and has two purposes:

1. The decoded control information is used to interrogate data conditions such as an adder carry, an adder sum, or a sign value, and to set a trigger called a stat to remember the results of the interrogation for future control action.
2. The status control information is also used to modify the controls specified by the data flow control segment of the ROS word. For example, the status control field is used to set a function register which will modify a data flow control field to cause data to be logically ANDed or ORed or EXCLUSIVE ORed depending on the value set into the function register. Figure 1 shows the mover function register.

The third control segment in the ROS word, called the emit field, is used as the source of a constant. It is used in conjunction with the data flow control segment for sign and zone insertion, as an adder input, and also for addressing the local storage.

The emit field provides a constant to the status control segment to set a function register and also to set a stat unconditionally.

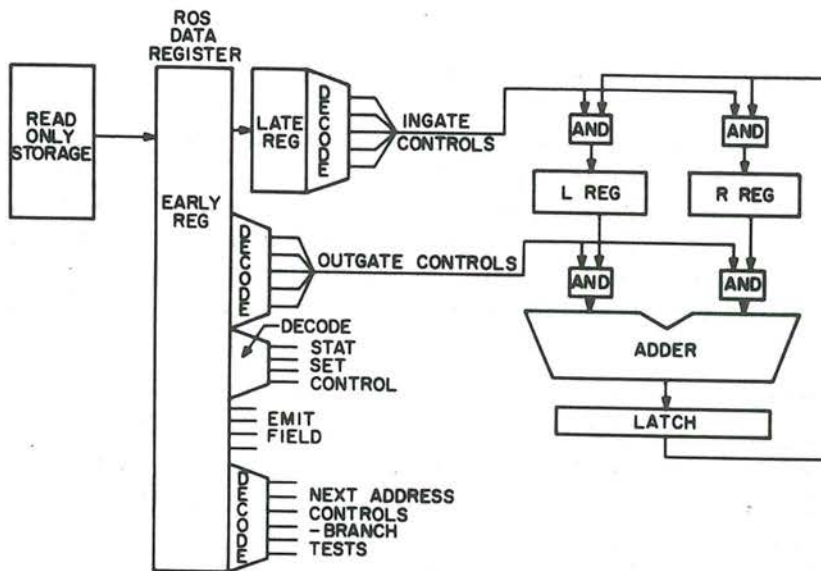


Figure 3. Relationships between ROS control word and data flow.

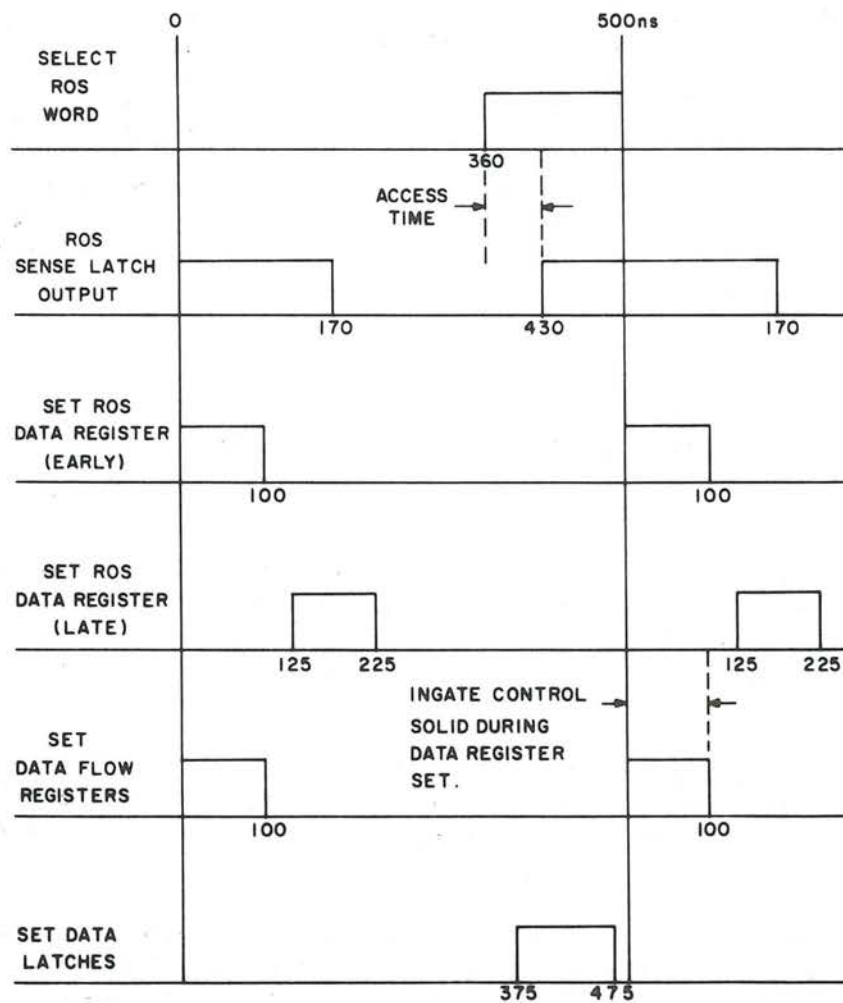


Figure 4. IBM System/360 Model 50 ROS and data flow timing.

of another machine or who microprogram special features, however, must work within the established framework and may make only minor modifications and additions to the control specifications. This is not so in a hardware-controlled machine.

It is therefore the burden of the original microprogrammer to specify the control fields and their micro-orders to best satisfy the performance objectives of the basic CPU while keeping the controls general enough to accommodate the future microprogramming of emulators and special features.

It is not practical to attempt to completely specify the ROS word format until the control designer has sequenced almost completely the execution of each program instruction and the instruction fetch and decode routine.

As a first pass at the control word specification, the basic data flow should be examined for independent data handling areas. On the Model 50 this is the adder path, the mover path, main storage and local storage. Each one of these areas is assigned several control fields for in-gating, out-gating and data manipulation. For example, the Model 50 adder path has the following control fields:

1. left adder input
2. right adder input
3. true or complement of left input
4. carry detection
5. shifter
6. adder latch to register

Next, micro-orders within a field are specified for those gates that are mutually exclusive and for combinations of gates which occur most frequently. See Figure 5. Since only one source may feed data into the right side of the adder at a time, the micro-orders controlling gates from R, M, and H are all mutually exclusive and placed in one control field.

As previously mentioned, a micro-order may activate more than one gate. For example, a Model 50 control field is used to decrement length counters  $G_1$  and  $G_2$ . The following micro-orders are in the same field:

- $DG_1$ , which decrements only counter  $G_1$ ;
- $DG_2$ , which decrements only counter  $G_2$ ; and
- $DG_1G_2$ , which decrements both counters  $G_1$  and  $G_2$ .

It is impossible to specify a micro-order for each gating combination likely to occur and still have a realizable ROS word width. When assigning a micro-order to control multiple gates, judgment must be used as to the use frequency of the combination and the time that will be sacrificed by doing the same job serially with successive microinstructions using simple micro-orders.

There are also several sign stats which hold the value of a fixed, floating-point, or decimal sign. In addition, byte stats are used to control the transfer of data from the CPU to main storage on a byte basis. See Figure 1.

An important function of the status control field is to set a function register for control of the data flow. The Model 50 has two function registers. One is a two-bit register which addresses one of the four local storage sectors for general-purpose, floating-point, working, or I/O registers.

The other function register is three bits wide, and controls data manipulation in the mover path. Six System/360 VFL instructions — move, move numeric, move zone, logical OR, logical AND, and logical EXCLUSIVE OR — are executed by one microprogram. The objective in each one of these instructions is to form a byte which is the function of two source bytes. If the source bytes are in the L and M registers (See Figure 1) and the resultant byte  $M_B^1$  is to be stored into M, then

$$M_B^1 = f(L_B, M_B)$$

One microprogram selects successive byte pairs, stores the resultant byte in the CPU and fetches more data from main storage as needed. The function  $f(L_B, M_B)$  is controlled by the function register which is set differently for each of the six instructions at the end of instruction preparation. Only one micro-order is needed to activate the function register for data manipulation. This technique has resulted in the saving of many microinstructions and five of the six micro-orders needed to directly specify the data function for these six System/360 instructions.

Another function of the status control field is the use of the emit field to inject a constant into the data flow as explained in the next section.

#### EMIT FIELD

Because the four-bit emit field on the Model 50 serves as a constant, it is the only ROS word segment which is not decoded. The emit field is used to:

1. set stats directly to the emit bit values, or on or off according to the one-bits present;
2. address a register in a sector of local storage;
3. enter a constant into the adder path;
4. enter a constant into the mover path.

The clever microprogrammer can choose working storage register addresses and general-purpose stats so that one emit value can satisfy several or all of the above functions in the same microinstruction.

An example of a microprogram is given in Figure 8. The microinstruction at ROS location 453 contains a branch micro-order BCARY which tests the carry out of the adder. BCARY is called an immediate branch because it tests a condition which is the result of action occurring in the present cycle. The Model 50 has 32 B-branch test micro-orders, but only two are immediate because of the time

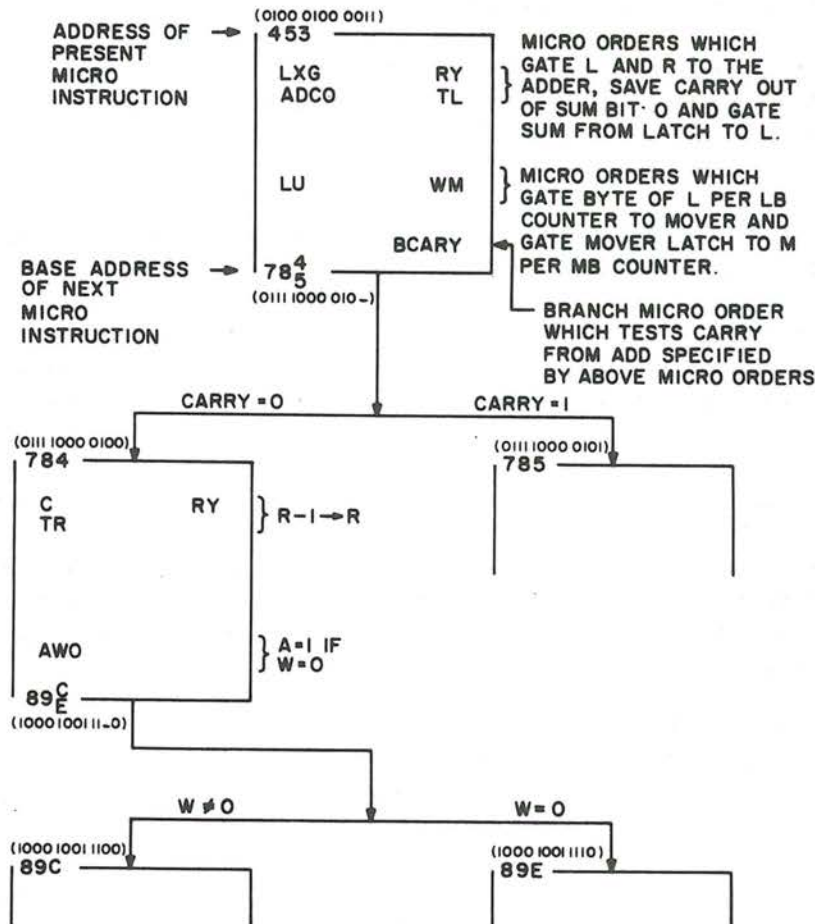


Figure 8. Flow chart representation of a microprogram.

required to decode the branch test and to form the ROS word address in time for selection of the next word.

The remaining thirty branch micro-orders test conditions in the immediately preceding cycle or conditions stored in stats, registers or counters.

The next microinstruction selected may be one of four by specifying another branch test forming the A-branch bit. There are 62 A-branch micro-orders in the Model 50.

While the simple branch tests are given high usage, some special branch tests were required on the Model 50 to handle the floating-point and the VFL instructions. By judicious selection of these branches a significant increase in performance and a saving of ROS words was realized.

The handling of VFL character instructions will serve as an illustration. These instructions are unique in that :

1. both operands are contained in main storage;
2. each operand may start on any byte boundary;
3. the operands may be as long as 256 bytes and contained in as many as sixty-five main storage words.

These instructions are handled by fetching the first words of operands one and two to the M and L registers respectively. See Figure 1. Byte counters, MB and LB, controlling the selection of bytes from M and L to the mover are initially set to the starting address of each operand. One microinstruction selects byte pairs, operates on them, and stores the resultant byte to M. It is desirable to know in the same cycle when either byte selected is the last available in the CPU for that operand and when the last byte pair is being handled. Thus an A branch condition is chosen which tests ( $G = 0$  or  $MB = 3$ ) and a B branch condition which tests ( $LB = 3$ ). See Figure 9.

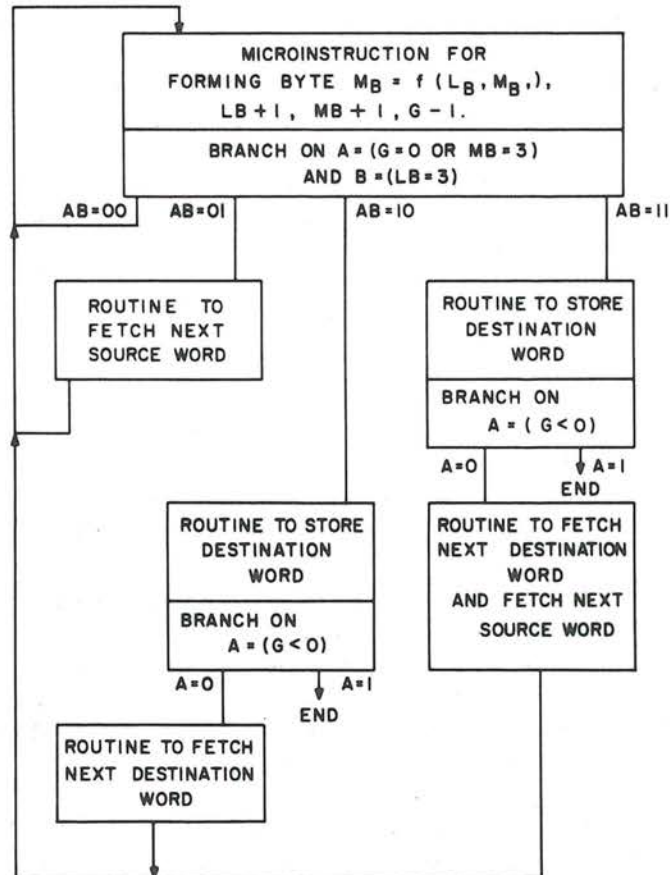


Figure 9. General program for byte handling of VFL character instructions.

in the Model 50 decreased the number of words because six instructions are executed by one microprogram. The use of special branches, such as the VFL branch as mentioned in the "Specification of Branch Test Micro-Orders" section, reduced the words and increased performance. The purist may consider the addition of such control hardware to supplement the ROS word controls as a deviation from basic ROS control philosophy. However, the objective is not to design an academically interesting machine but instead to produce a high performance, low-cost processor justifying some compromises to achieve this goal.

The attainment of the best control design depends upon the objectives and many restrictions of a particular project. Control design is still an art limited by the creativity of the designer.

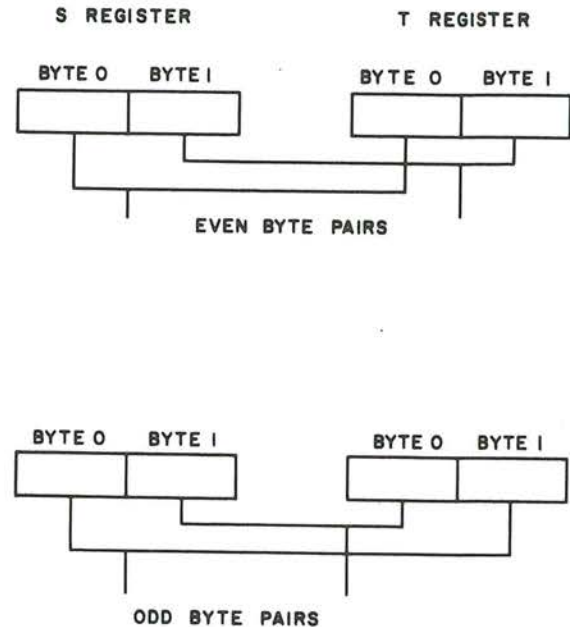


Figure 10. Even and odd byte pairs.

## REFERENCES

1. G.A. Blaauw, F. P. Brooks, Jr. "The Structure of System/360" Part I -- "Outline of Logical Switches," IBM Systems Journal, Vol. 3, #213.  
W. Y. Stevens, "The Structure of System/360" Part II -- "System Implementations," IBM Systems Journal, Vol. 3, #213.
2. L. J. Boland, "Analysis of Read-Only Memory for Control," Thesis for M. E. E. degree at Syracuse University, June 1963, IBM TIC 63E6-4362.