Technical Report 68-78 September, 1968

A Higher-order Language for Describing Microprogrammed Computers

by
Yaohan Chu, Professor of Computer Science
and Electrical Engineering

UNIVERSITY OF MARYLAND
COMPUTER SCIENCE CENTER
COLLEGE PARK, MARYLAND

Technical Report 68-78 September, 1968

A Higher-order Language for Describing Microprogrammed Computers

by

Yaohan Chu, Professor of Computer Science and Electrical Engineering

This work was supported by Grant (NsG 398) of the National Aeronautics and Space Administration to the Computer Science Center of the University of Maryland.

NGL-21-002-008

Table of Contents

A Higher-order Language for Describing Microprogrammed Computers

Abstract

This report discusses the application of a higher-order language for describing microprogram controlled computers. The language is symbolic, algorithmic and descriptive. It is concise and precise in describing computer elements, micro-operations, sequences and microprograms. A simulator has been constructed for simulating microprogrammed computers described in this language.

The language is introduced by describing a simple sequence, first in conventional sequential logic control and then in microprogram control. A simple, microprogram controlled computer is then described. The description shows the configuration, the microprogram itself. Results of the simulation of these examples are partly shown.

A Higher-order Language for Describing Microprogrammed Computers

1. Introduction

The idea of microprogramming (or microprogrammed Computer or microprogram controlled computers) was first conceived in 1951 (1, 2, 5, 8). The early microprogrammed computers were mostly designed and constructed to make one computer to behave like several. Today, most of the computers manufactured are microprogrammed computers. However, microprogrammed computers of today are designed and constructed to make a family of computers to behave like one. In either case, it is an application of the original idea that the sequencing of the built-in micro-operations can be changed by changing the microprogram.

Microprogramming is an art of specifying the controls of all the built-in micro-operations into sequences for the purpose of performing complex logical, arithmetic and other functions. The ideas of loops, subroutines, multiprocessing and multiprogramming in the conventional programming can be and have been applied in microprogramming.

Recently, there has been a growing, renewed interest in microprogramming. One important reason of this renewed interest is that the microprogram in a microprogrammed computer becomes a bridge between the software and hardware designers. By means of this bridge, behavior of a microprogrammed computer can be, to some extent, altered (10) without resorting to hardware changes.

A microprogram is an array of 1's and 0's. Like the preparation of programs of early digital computer, preparation of a microprogram is fraught with error. When it is prepared, the meaning of the array of 1's and 0's is difficult to be understood. Therefore, the idea of describing the microprogram by a higher-order algorithmic language and generating the microprogram by compilation is applied. This report applies a higher-order algorithmic language (16) to the description of microprogrammed computer. A simulator has been constructed and is now available (22). Once the microprogrammed computer is described by the higher-order language, it is punched into a deck of cards and the paper computer is ready for simulation.

We shall first introduce this higher-order language by describing a serial complement sequence. Then, a microprogrammed computer is described by this higher-order language.

2. A Serial Complement Sequence

This sequence serially complements every bit of a word stored in a shift register. This sequence is first described by sequential logic control and then by micro-programmed control.

2.1 Sequential logic control

A configuration for carrying out this complement sequence is shown in Figure 1. Register A is the shift register when the word is stored. When register A is being shifted one bit to the right, the contents of bit A(1) at the right end of the register are complemented by a logical NOT block and then transferred to bit A(10) at the left end of the register. After complementing and right-shifting in this manner ten times, each bit of register A is complemented. Counter C counts the number of times. Counter T and clock P generate the control signals. Switch START initializes the necessary operations for the sequence, and light FINI indicates the completion of the operation. This configuration can also be described by the following statements:

```
     Register,     A(10-1),          $shift register
                   T(1-3),           $control register      (1)
                   C(0-3).           $counter
     Switch,       START(ON)         $start switch
     Light,        FINI(ON,OFF)      $completion indicator
     Clock,        P
```

The sequential operations of the complement sequence implemented by the configuration of Figure 1 can be described by the sequence chart shown in Figure 2. As shown, when switch START is turned to the ON position, counter C is reset to 0 and light FINI is set to the OFF condition. Then, the complementing and right-shifting operation is performed and at the same time counter C is incremented by 1. Counter C is next tested for a value of 10. If the value is not 10, the complementing and right-shifting operation and counter incrementing operation are again performed. Counter C is again tested. This process continues on until counter C reaches a value of 10; by then, each bit of register A is complemented. Light FINI is then turned to the ON condition and the sequence is terminated.

The above described sequential operations can be described by the following statements:

```
/START(ON)/  C←—0, FINI←—OFF, T←—100
/T(1)*P/     A←—A(1)'-A(10-2), C←—countup C, T(1,2)←—01 (2)
/T(2)*P/     IF(C=10) THEN (T(2,3)←—01) ELSE (T(1,2)←—10)
/T(3)*P/     FINI←—ON
             END
```

In the above description, counter T is a ring counter for generating the control sequence. When bit T(1) is 1 (but not other bits of register T), micro-operations for complementing, shifting and incrementing are performed. When bit T(2) is 1, counter C is tested and sequence branched. When T(3) is 1, light FINI is turned to the ON condition. Since there is no micro-operation to change the contents of register T when T(3) is 1, the sequence will continue to execute to turn light FINI to the

ON condition each time the clock signal occurs.

It should be noted from the above description that, excluding the micro-operations to be activated manually by switch START and the micro-operations involving with register T, there are four micro-operations as listed below:

$$\begin{aligned} &A \longleftarrow A(1)'\text{-}A(10\text{-}2) \\ &C \longleftarrow \text{countup } C \\ &\text{FINI} \longleftarrow \text{ON} \\ &\text{IF } (C=10) \text{ THEN } (.....) \text{ ELSE } (.....) \end{aligned} \qquad (3)$$

As will be shown, more micro-operations are required in the case of microprogrammed control.

2.2 Microprogram control

The sequential operations of the above complement sequence may also be controlled by means of a microprogram stored in a control memory. Let the control memory be named CM; its address register and buffer register be named H and F respectively. Figure 3 shows a configuration for executing a serial complement sequence by microprogrammed control which is described by the statements

```
    Register,  A(10-1)          $shift register
               C(0-3),          $counter              (4)
               H(0-2),          $address register
               F(0-8),          $buffer register
    Memory     CM(H)=CM(0-7,0-8) $control memory
```

Switch,	START(ON)	$start switch
Light,	FINI(ON,OFF)	$completion indicator
Clock,	P(1-2)	$clock source

The above configuration replaces control Register T in the configuration of Figure 2 by Control Memory CM and registers H and F. A two-phase clock is used. They require three micro-operations as listed below:

```
F ← CM(H)                                              (5)
H ← countup H
IF (C=10) THEN (H ← countup H) ELSE (H ← F(0-2))
```

The first micro-operation reads a word from the control memory to the buffer register F. The second micro-operation increments register H by 1. The last one is a conditional micro-operation which tests and branches the sequence; this micro-operation replaces the similar one in the statements (3). There are six micro-operations altogether.

Each word in the control memory is called a control word (also called a micro-instruction) which consists of 9 bits. The format of the control word is shown in Figure 4. Bits F(0-2) is an address field for the H register. Each of the remaining six bits is assigned to one of the six micro-operations. When bit F(j) where j is 3 through 8 is 1, the control signal for the jth micro-operation is commanded. Since more than one F(j) bits can be 1, more than one micro-operations are possible. This is one way that parallel micro-operations are described. Several examples of the control word are shown in Figure 5. The commas in Figure 5 serve the purpose of better readability and they do not actually exist. These three control words, as will be described, actually constitute the microprogram for the complement sequence.

The sequential operations of the complement sequence implemented by the configuration in Figure 3 are described by the statements:

```
Comment, start the complement sequence
/START(ON)/   H←0, C←0, FINI←OFF, F←040        (6)
/F(3)*P(2)/   F←CM(H)
comment, complement and shift
/F(7)*P(1)/   A←A(1)'-A(10-2)
/F(6)*P(1)/   C←countup C
/F(5)*P(1)/   H←countup H
/F(3)*P(2)/   F←CM(H)
Comment, test and branch
/F(8)*P(1)/   IF (C=10) THEN (H←countup H) ELSE (H←F(0-2))
/F(3)*P(2)/   F←CM(H)
Comment, completion indication
/F(4)*P(1)/   FINI←ON
              END
```

The above description is divided into four groups. Each group is headed by a comment statement and carries out the micro-operations in each of the four blocks in the sequence chart of Figure 2. The first comment statement refers to a group of micro-operations, described by the two statements directly below this comment statement, to initialize the execution of the sequence when switch START is turned to the ON position. These micro-operations reset registers H and C, turn light FINI to the OFF condition, and set register F to 040_8 to command the micro-operations of reading a word out of the control memory. The second comment statement refers to the four micro-operations described by the four statements directly below this comment statement; this group of micro-operations are those specified by the first control word 056_8 in the control memory shown in Figure 5. This control word consists of four 1's each of which commands one of the four micro-operations (complement and shift, increment C, increment H and read the memory). The third comment statement refers to the micro-operations specified by the second control word 041_8 in the control memory. This control word consists of two 1's each of which commands one of the two micro-operations (test-branch and read the memory). The last comment statement refers to the micro-operation to turn on light FINI as specified by the third control word 020_8 in the control memory. Thus, statements (6) describe the operations of the sequence specified by the control words in the microprogram.

The execution of the microprogram is described by the sequence chart in Figure 6. In this chart, there exists a loop in which there are as many branches as the number of micro-operations. During each cycle of the loop, one or more micro-operations specified by the control word are executed during the first clock phase, and another control word is read out of the memory during the second clock phase. The next control word is located either by incrementing register H or by transferring the address in bits F(0-2) to the H register. The speed in executing the loop is at present one serious limitation on the speed of microprogram controlled computers.

3. A Micro-program Controlled Computer

Having introduced the language for describing microprogram control logic, we now proceed to describe a simple, microprogram-controlled, stored-program computer.

3.1 Configuration

The configuration of the computer is shown in Figure 7. The computer has a random access memory M with address register C and buffer register R. The memory has a capacity of 4096 words and each word has 18 bits. There are program register D, arithmetic register A, start-stop control register G and switches START, STOP and POWER. For microprogram control, there is control memory CM with address register H and buffer register F. This configuration is described by the following statements:

```
    Register,      R(0-17),             $buffer register for memory M (7)
                   A(0-17),             $arithmetic register
                   C(0-11),             $address register for memory M
                   D(0-11),             $program register
                   F(1-18),             $buffer register for memory CM
                   H(1-5),              $address register for memory CM
                   G                    $start-stop control register
    Subregister,   R(OP)=R(0-5),        $op-code part of register R
                   R(ADDR)=R(6-17),     $address part of register R
                   F(ADDR)=F(1-5)       $address part of register F
    Memory,        M(C)=M(0-4095,0-17)  $main memory
                   CM(H)=CM(0-31,1-18)  $control memory
```

Switch,	POWER(ON),	$power switch
	START(ON),	$start switch
	STOP(ON),	$stop switch
Clock,	P(1-3)	$three-phase clock

As shown above, the clock P is of three phases. Register G is of single bit.

3.2 Description of the sequential operations

The instruction format consists of a 6-bit op-code field and a 12-bit address field. The instruction set consists of instructions CLEAR-ADD, ADD, SUBTRACT, STORE, JUMP, JUMP-ON-MINUS, SHIFT-RIGHT-ONE-BIT, CIRCULAR-LEFT-SHIFT-ONE-BIT, and STOP. Each instruction is implemented by a sequence similar to the previously described complement sequence. These sequences are called execution sequences. For example, the steps for the ADD sequence are:

```
step 1,  R ⟵ M(C)                                    (8)
     2,  A ⟵ A add R,  H ⟵ F(ADDR)
     3,  C ⟵ D, F ⟵ CM(H)
```

In addition, a fetch sequence is required. The steps for the fetch sequence are:

```
step 1,  R ⟵ M(C)                                    (9)
     2,  H ⟵ R(OP), C ⟵ R(ADDR), D ⟵ countup D
     3,  F ⟵ CM(H)
```

When all these sequences are specified, the format of the control word is chosen, and the control function for each bit of the control word together with the clock phase is next assigned to form the control signals. The format of the control word and the assignment of the control function are shown in the upper portion of Figure 8. With these control signals, the sequential operations of the computer can now be described by the statements:

```
Comment, start computer operation                               (10)
    /POWER(ON)/   G←--0, F←--0, H←--0, C←--0, D←--0
    /START(ON)/   G←--1, F(8)←--1
    /STOP(ON)/    G←--0
COMMENT, FETCH SEQUENCE WHEN H=0
    /F(6)*P(1)/   R←--M(C)
    /F(7)*P(2)/   H←--R(OP), C←R(ADDR), D←--countup D
    /F(8)*P(3)/   IF(G)THEN(F←--CM(H))ELSE(H←--0, C←--0, D←--0, R←--0)
COMMENT, ADD SEQUENCE WHEN H=1
    C/F(6)*P(1)/  R←--M(C)
     /F(11)*P(2)/ A←--A ADD R
     /F(9)*P(2)/  H←--(ADDR)
     /F(9)*P(3)/  C←--D, F←--CM(H)
COMMENT, SUBTRACT SEQUENCE WHEN H=2
    C/F(6)*P(1)/  R←--M(C)
     /F(12)*P(2)/ A←--A SUB R
    C/F(9)*P(2)/  H←--F(ADDR)
    C/F(9)*P(3)/  C←--D, F←--CM(H)
COMMENT, JUMP-ON-MINUS SEQUENCE WHEN H=3
     /F(14)*P(1)/ IF (A(0))THEN(D←--R(ADDR))
    C/F(9)*P(2)/  H←--F(ADDR)
    C/F(9)*P(3)/  C←--D, F←--CM(H)
```

```
COMMENT, STORE SEQUENCE WHEN H=4
     /F(10)*P(1)/   R←--A
     /F(10)*P(2)/   M(C)←--R
    C/F(9)*P(2)/    H←--F(ADDR)
    C/F(9)*P(3)/    C←--D, F←--CM(H)
COMMENT, JUMP SEQUENCE WHEN H=5
     /F(13)*P(1)/   D←--R(ADDR)
    C/F(9)*P(2)/    H←--F(ADDR)
    C/F(9)*P(3)/    C←--D, F←--CM(H)
COMMENT, SHIFT RIGHT ONE-BIT SEQUENCE WHEN H=6
     /F(15)*P(1)/   A←--A SHR A
    C/F(9)*P(2)/    H←--F(ADDR)
    C/F(9)*P(3)/    C←--D, F←--CM(H)
COMMENT, CIRCULAR SHIFT LEFT ONE-BIT WHEN H=7
     /F(16)*P(1)/   A←--A CIL A
    C/F(9)*P(2)/    H←--F(ADDR)
    C/F(9)*P(3)/    C←--D, F←--CM(H)
COMMENT, CLEAR ADD SEQUENCE WHEN H=8
    C/F(6)*P(1)/    R←--M(C)
     /F(17)*P(1)/   A←--0
    C/F(11)*P(2)/   A←--A ADD R
    C/F(9)*P(2)/    H←--F(ADDR)
    C/F(9)*P(3)/    C←--D, F←--CM(H)
COMMENT, STOP SEQUENCE WHEN H=9
     /F(18)*P(1)/   G←--0
    C/F(9)*P(2)/    H←--F(ADDR)
    C/F(9)*P(3)/    C←--D, F←--CM(H)
                    END
```

In the above description, there is a letter C which occurs at the first position of some of the statements. The presence of letter C at the first position of a statement makes the statement a comment statement. Those statements with letter C are actually not needed as far as the completeness of the description is concerned. The insertion of these redundant statements makes the description of the microprogram more comprehensible.

3.3 Microprogram

As shown, the above description is divided into eleven groups. Each group is headed by a comment statement and carries out the operations specified for a sequence. One group is for a start sequence, another for a fetch sequence, and each of the remaining group is for one of the nine execution sequences for the nine instructions. Except the micro-operations in the first group which are controlled manually, the micro-operations in each group are controlled by one control word; this is possible for the ten sequences under the microprogram control. The ten control words in octal numbers are shown in Table 1.

In examining the above statement description of the computer, there are 22 micro-operations. If one control bit is assigned for each micro-operation, as in a manner shown in Figure 4, the control word would require 22 control bits plus address bits or a total of 27 bits. For a practical computer, the number of micro-operations will be much larger, and the word length for the control word will be much too long. Therefore, it is desirable to find ways to reduce the length of the control word. A few is to be mentioned here. However, the reduction so achieved is at the expense of flexibility of microprogram control.

Table 1, Microprogram

Control Memory Address	Control Words (Octal numbers)
0	016 000
1	011 200
2	011 100
3	001 020
4	001 400
5	001 040
6	001 010
7	001 004
8	011 202
9	001 001

Instead of a total of 27 bits, the control word in Figure 7 has a word length of 18 bits. This reduction is achieved by grouping micro-operations under one control signal. For example, as shown in Figure 8, three micro-operations are grouped under control signal F(7)*P(2); this is possible because they occur at the same time. But this reduction gives up the use of each individual micro-operation when such a use occurs. The reduction is also achieved by grouping under one control bit those micro-operations which can take place in the phases of one clock cycle. For example, as shown in Figure 8, micro-operation H<---E(ADDR) at clock phase P(2) and micro-operations C<---D and F<---CM(H) both at clock phase P(3) can be grouped under control bit F(9). This reduction again limits the individual use of each of these micro-operations.

Another possible way of reducing three control bits of the control word in Figure 8 is as follows: The six control bits F(13-18) are replaced by three bits F(13-15) and a decoder which is attached to these three bits. Let the decoder be specified by the decoder statement,

Decoder, K(0-5)=F(13-15)

The six terminals of the decoder are K(j) where j is equal to 0 through 5. The six new control signals are: K(0)*P(1)......., K(5)*P(1). This reduction is possible because only one of the six micro-operations occurs at any one time. But the use of decoder limits the possibility of using more than one of these six micro-operations at the same time when such a need arises.

4. Simulation

A simulator which accepts the computer organization described in the CDL has been constructed (2). Statements (1) and (2) for the serial complement sequence by sequential logic control are punched into a deck cards, and the listing is shown in Figure 9. Statements (4) and (6) for the serial complement sequence by microprogram control are similarily punched and listed in Figure 11. Statements (7) and (10) for the microprogram controlled computer are punched and listed in Figure 13. Sample outputs of the simulation runs for the three simulations are shown in Figures 10, 12 and 14 respectively. The reader can trace the contents of the registers and other elements clock by clock and statement by statement.

The first eleven lines in Figure 9 and the first twelve lines in Figure 11 are control cards for the translation phase of the simulation. The last ten lines in Figures 9 and 11 are control cards for the execution phase of the simulation. In this execution phase, the desired output from the printer is specified. For example, the *OUTPUT statement in Figure 9 specifies that the contents of the registers A, T, C and FINI are printed at every clock. The outputs shown in Figures 10, 12 and 14 are the results of such output statements.

The statements in the listings of Figures 10, 12 and 14 have slight differences from statements (1), (2), (4), (6), (7), and (10). These differences are enumerated below.

(a) All letters have to be capital letters

(b) Subscripts are octal number instead of decimal numbers.

(c) Comments such as $buffer register...at the end of a statement are not permitted.

(d) First position of statement is not used except a letter C to indicate the statement is a comment statement.

(e) Operators ←--, =, countup, cil, shr, add, and sub are replaced respectively by =, .EQ., .COUNT., .SHR1., .ADD. and .SUB..

It is apparent that these differences are due to the limited character set of the key punch or to the convenience in the construction of the simulator.

In Figure 13, the last eleven cards are the control cards for the execution phase of the simulation. Notice the *LOAD card which loads a test program in the language of the descripted computer into the memory. This is a machine language program. Both this program and its symbolic program are shown in Table 2. This test program includes all the available nine instructions of the computer.

Table 2, Test Program

Memory Address (decimal)	Memory Word Contents (octal)	Symbolic Program
0	100 100	CLA 64
1	010 101	ADD 65
2	040 102	STO 66
3	070 000	CLS 0
4	030 006	JOM 6
5	050 003	JMP 3
6	040 103	STO 67
7	020 102	SUB 66
8	060 060	SHR 0
9	110 000	STP 0
..	....	
..	....	
64	000 001	1
65	100 001	-31
66	000 000	
67	000 000	

5. References

1. M. V. Wilkes, "The best way to design an automatic calculating machine", Manchester University Computer Inaugural Conference, July 1951, pp. 16-18.

2. M. V. Wilkes and J. B. Stringer, "Microprogramming and the design of the control circuits in an electronic digital computer", Proceedings of the Cambridge Philosophical Society 49, Part 2, 1953, pp. 230-238.

3. H. T. Glantz, "A note on microprogramming", Journal of the Association for Computing Machinery 3, No. 2, April 1956, pp. 77-84.

4. R. J. Mercer, "Micro-programming", J. of the ACM, April 1957, pp. 157-171.

5. M. V. Wilkes, "Microprogramming", Proceedings of the Eastern Joint Computer Conference, December 1958, pp. 18-20.

6. J. V. Blankenbaker, "Logically microprogrammed computers", IRE Transactions on Electronic Computer EC-7, June 1958, pp. 103-109.

7. G. P. Dineen, I. L. Lebow, I. S. Reed, "The logic of CG24", Proceedings of the Eastern Joint Computer Conference, December 1958, pp. 91-94.

8. M. V. Wilkes, W. Renwick, and D. J. Wheeler, "The design of the Control Unit of an Electronic Digital Computer", Proceedings of the IEE (London), vol. 105, pt. B, June 1959, pp. 121-128.

9. T. W. Kampe, "The design of a general-purpose microprogram-controlled computer with elementary structure", IRE Transactions on Electronic Computers EC-9, June 1960, pp. 208-213.

10. H. M. Semarne and R. E. Porter, "A stored logic-computer", Datamation 2, No. 5, May 1961, pp. 33-36.

11. A. Graselli, "The design of program-modifiable microprogrammed control units", IEEE Transactions on Electronic Computers EC-11, June 1962, pp. 336-339.

12. H. Hagiware, "The KT Pilot Computer - A Microprogrammed computer with a Phototransitor fixed memory", Proceedings of AFIPS Congress, 1962, pp. 318-321.

13. G. B. Gerace, "Microprogrammed control for computing systems", IEEE Transactions on Electronic Computers, December 1963, pp. 733-747.

14. E. Boutwell, Jr. and E. A. Hoskinson, "The Logical Organization of the PB 440 Microprogrammable computer", Proceedings of the FJCC, 1963, pp. 201-213.

15. S. G. Tucker, "Emulation of large systems", Communications of the Association for Computing Machinery 8, No. 12, December 1965, pp. 753-761.

16. Y. Chu, "An Algol-like Computer Design Language", Communications of the ACM, Oct. 1965, pp. 607-615.

17. S. Tucker, "Microprogram Control for System 360", IBM Systems Journal, Vol. 6, No. 4, 1967, pp. 222-241.

18. H. Weber, "A Micro-programmed Implementation of EULER on IBM system/360 Model 30", Communications of the ACM, September 1967, pp. 549-558.

19. B. D. McCurdy and Y. Chu, "Boolean Translation of a Macro Logic Design", Digest of the first annual IEEE Computer Conference, Sept. 6-8, 1967, pp. 124-127.

20. Y. Chu and C. K. Mesztenyi, "Macro Logic Design of Digital Computers", Technical Report 67-57, Computer Science Center, University of Maryland, November, 1967.

21. D. M. Fredrick, "Simulation of a Large scale general-purpose computer", Master Thesis, Department of Electrical Engineering, University of Maryland, June, 1968.

22. C. K. Mesztenyi, "Computer Design Language, Simulation and Boolean Translation", Technical Report 68-72, Computer Science Center, University of Maryland, June, 1968.

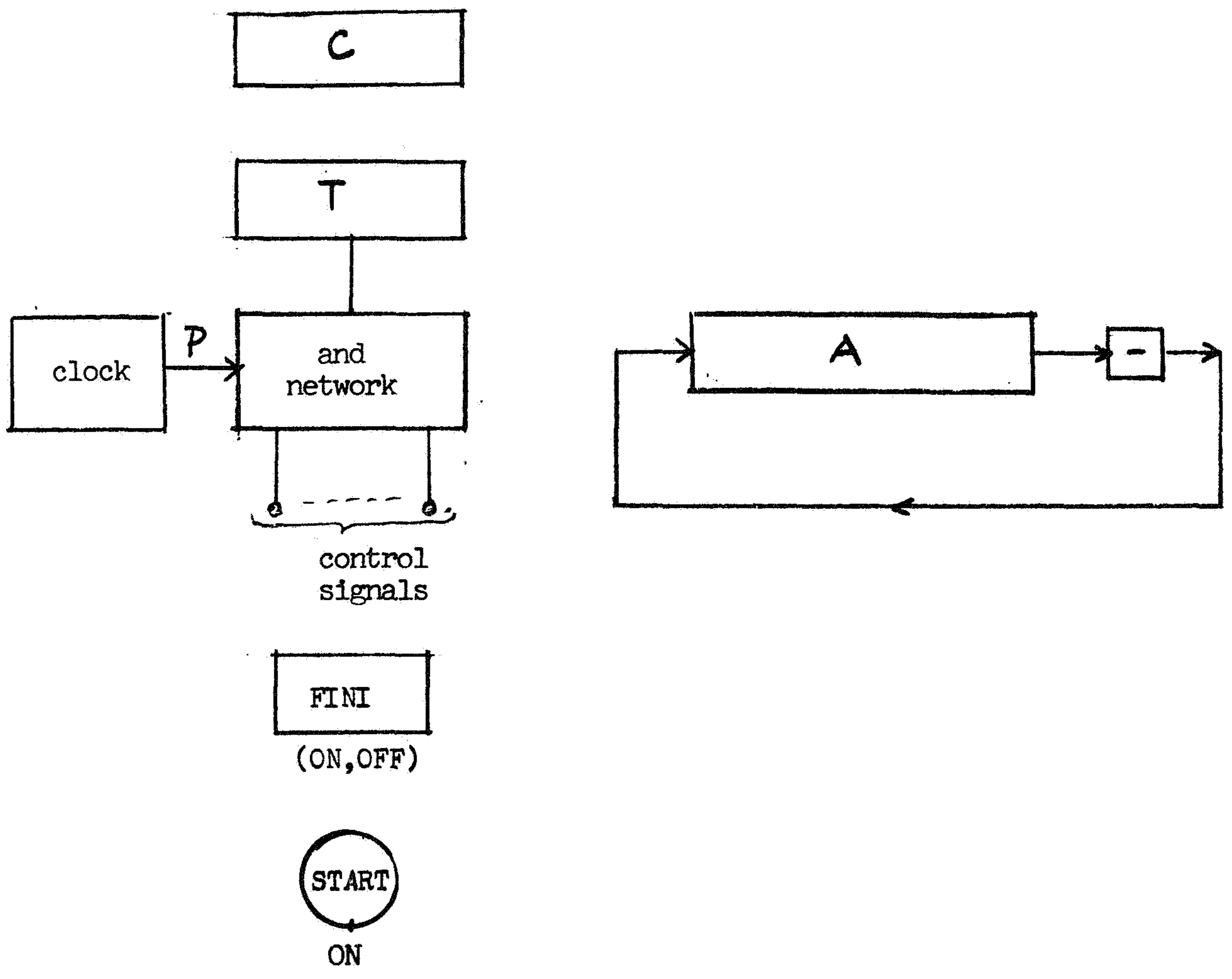

Fig. 1 Configuration for a serial complement sequence by sequential logic control

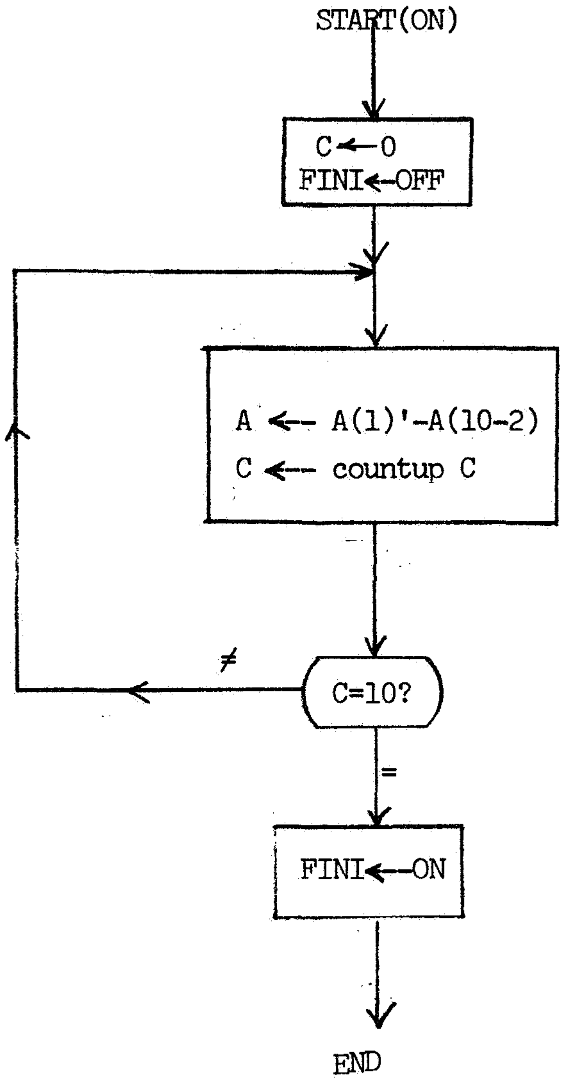

Fig. 2. sequence chart for a serial complement sequence

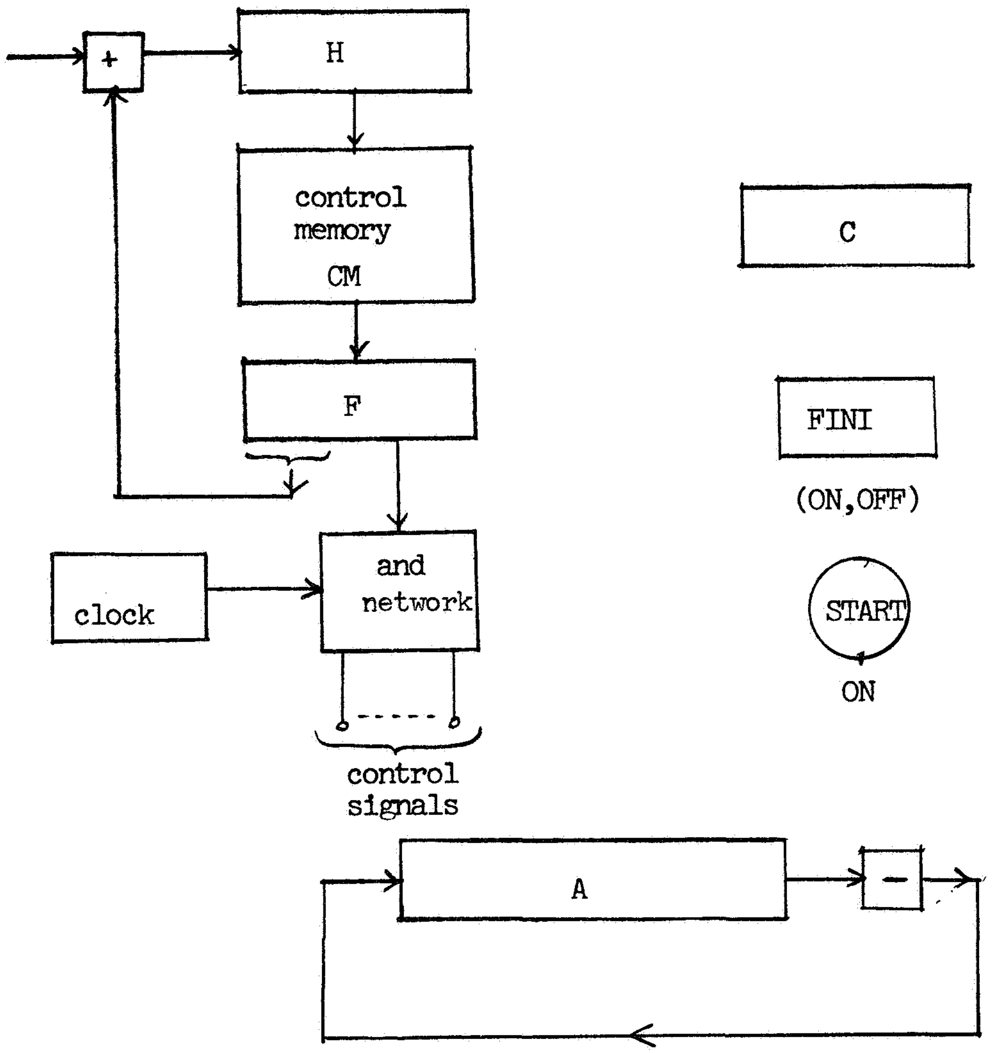

Fig. 3 Configuration for a serial complement sequence by microprogrammed control

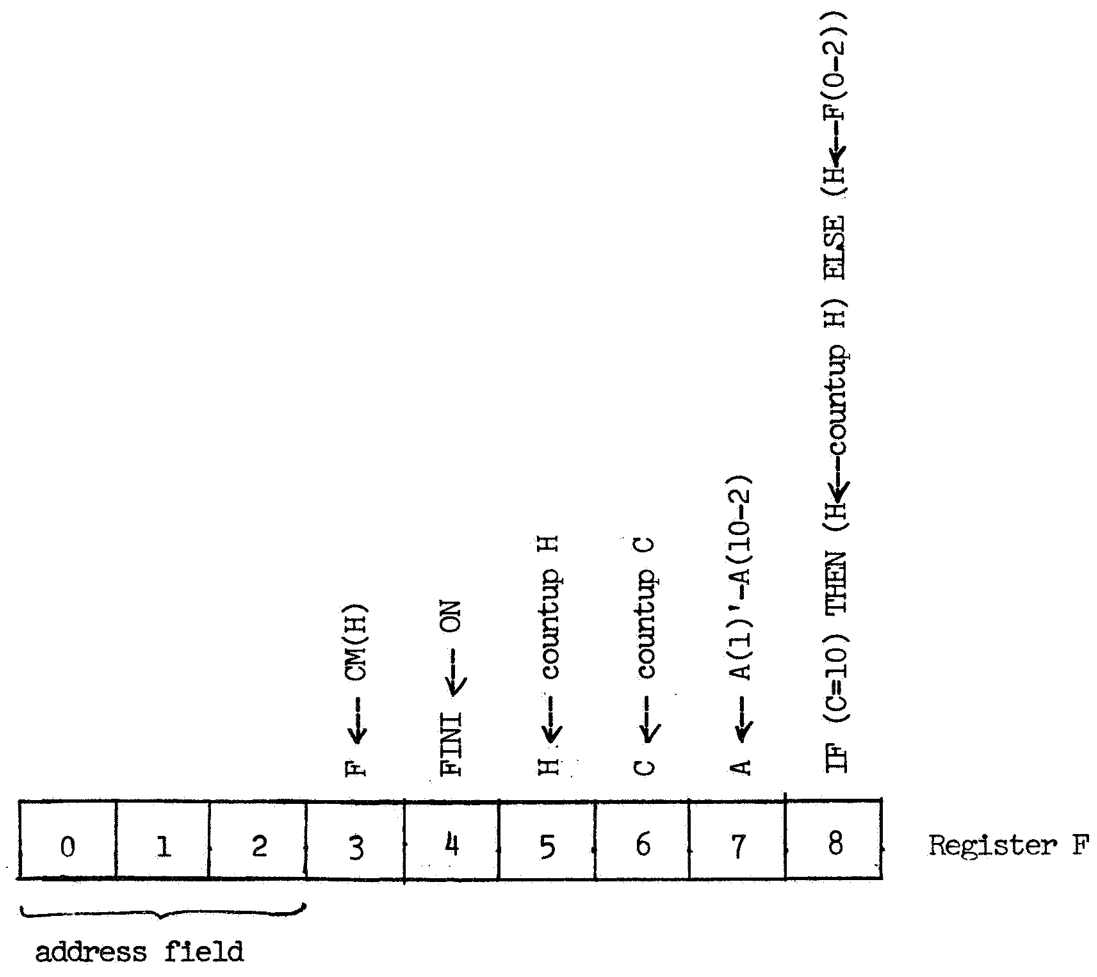

Fig. 4 Control word format

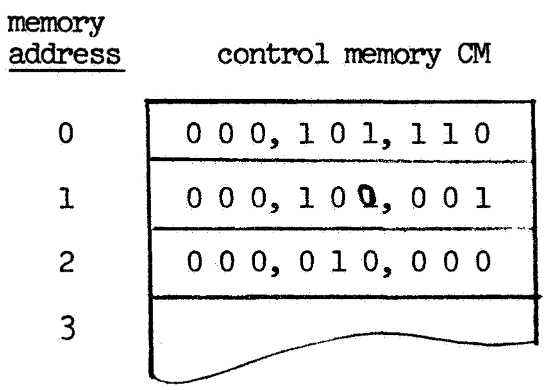

Fig. 5, Microprogram in the control memory

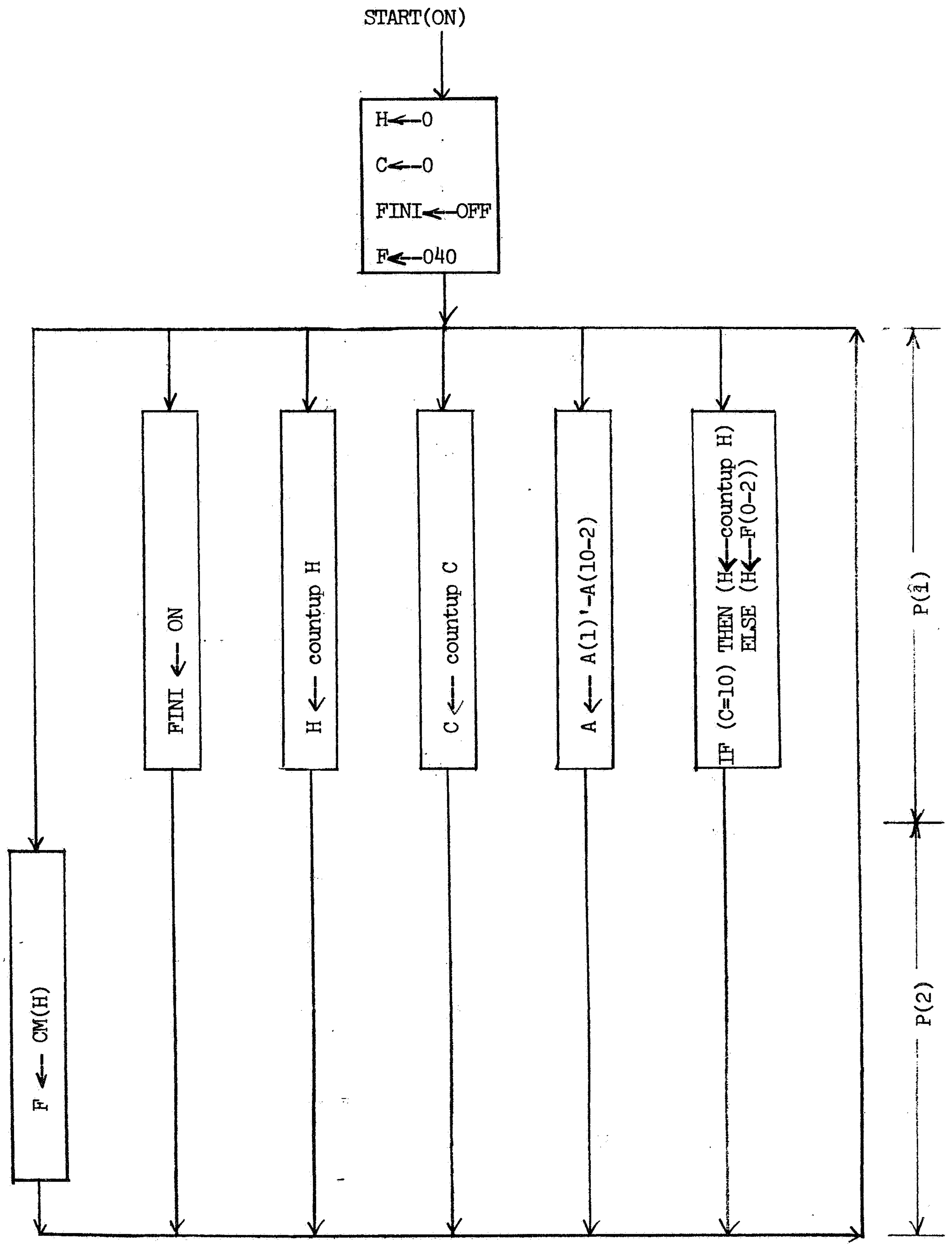

Fig. 6 Sequence chart for the serial complement sequence

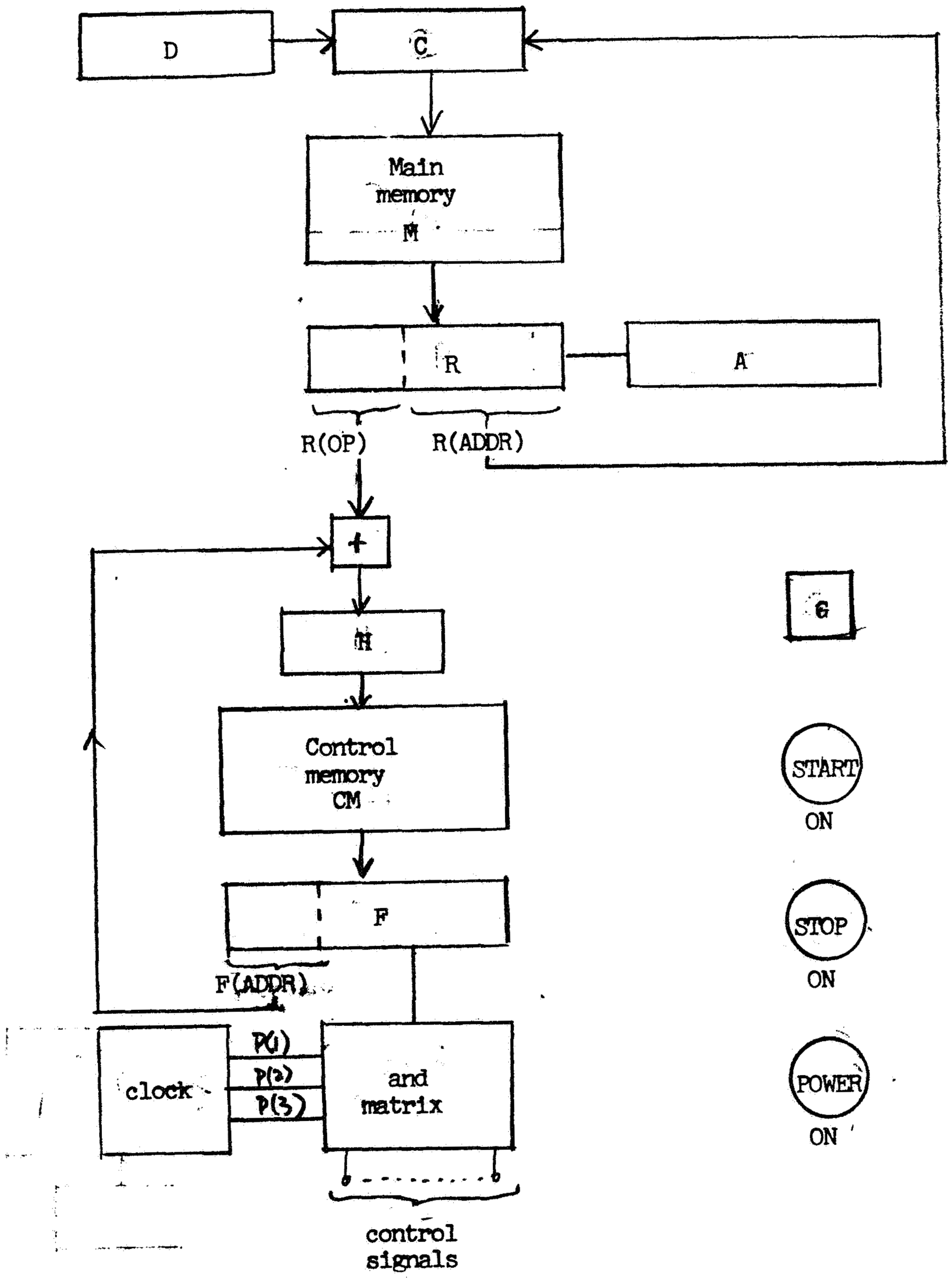

Fig. 7, Configuration for a simple stored-program computer

		Micro-operations	R←M(C)	H←R(OP), C←R(ADDR), D←countup D	IF(G)THEN (F←CM(H)) ELSE(H←0,C←0,D←0, R←0)	H←F(ADDR)	C←D, F←CM(H)	R←A	M(C)←R	A←A add R	A←A sub R	D←R(ADDR)	If(A(O))THEN (D←R(ADDR))	A←shr A	A←cil A	A←0	G←0
		Clock	P(1)	P(2)	P(3)	P(2)	P(3)	P(1)	P(2)	P(2)	P(2)	P(1)	P(1)	P(1)	P(1)	P(1)	P(1)
H	Code	F(1-5)	F(6)	F(7)	F(8)	F(9)		F(10)		F(11)	F(12)	F(13)	F(14)	F(15)	F(16)	F(17)	F(18)
0	FETCH	00000	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	ADD	00000	1	0	0	1	1	0	0	1	0	0	0	0	0	0	0
2	SUB	00000	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0
3	JOM	00000	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0
4	STO	00000	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
5	JMP	00000	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0
6	SHR	00000	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0
7	CLS	00000	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0
8	CLA	00000	1	0	0	1	1	0	0	1	0	0	0	0	0	1	0
9	STP	00000	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1

Note, H = control memory address
F(1-5) = next control memory address
F(6-18) = control bits

Fig. 8, The control word format and the microprogram.

```
$IBSYS
$*     MOUNT TAPE 2987 ON A9, RING OUT
$*     SAVE
$PAUSE
$ATTACH           A9
$AS               SYSLB4
$REWIND           SYSLB4
$EXECUTE          USER
$ID    OUYANG, B.,*001/01/125*2M*100P*T$
$CDL3
$TRANSLATE
*MAIN
C
COMMENT    ****A SERIAL COMPLEMENT SEQUENCE
C
 REGISTER,            A(12-1), T(1-3), C(0-3)
 REGISTER,            FINI
 SWITCH,              START(ON)
 CLOCK,               P
 /START(ON)/    T=4, FINI=0, C=0
 /T(1)*P/       A(12-1)=A(1)'-A(12-2), C=C.COUNT., T(1,2)=1
 /T(2)*P/       IF (C.EQ.12) THEN (T(2,3)=1) ELSE (T(1,2)=2)
 /T(3)*P/       FINI=1
 END
$SIMULATE
*OUTPUT    CLOCK(1)=A,T,C,FINI
*SWITCH    1,START=ON
*LOAD
     A=7C'
*SIM       30,3
*RESET     CYCLE
'
$IBSYS
$RESTORE
```

Fig. 9 Listing of the description of a serial complement sequence in sequential logic control

CUTPUT CF SIMULATION

```
SWITCH INTERRUPT
   START  = CN
       A = ..C7C7          T = .....4          C = ....CC        FINI = .....C
*****************************************************
LABEL CYCLE    1                  TRUE LABELS                   CLOCK TIME =    1
                                /T(1)*P/
       A = ..0343          T = .....2          C = ....C1        FINI = .....C
*****************************************************
LABEL CYCLE    2                  TRUE LABELS                   CLOCK TIME =    2
                                /T(2)*P/
       A = ..C343          T = .....4          C = ....C1        FINI = .....C
*****************************************************
LABEL CYCLE    3                  TRUE LABELS                   CLOCK TIME =    3
                                /T(1)*P/
       A = ..C161          T = .....2          C = ....C2        FINI = .....C
*****************************************************
LABEL CYCLE    4                  TRUE LABELS                   CLOCK TIME =    4
                                /T(2)*P/
       A = ..C161          T = .....4          C = ....C2        FINI = .....C
*****************************************************
LABEL CYCLE    5                  TRUE LABELS                   CLOCK TIME =    5
                                /T(1)*P/
       A = ..CC70          T = .....2          C = ....C3        FINI = .....C
*****************************************************
LABEL CYCLE    6                  TRUE LABELS                   CLOCK TIME =    6
                                /T(2)*P/
       A = ..CC70          T = .....4          C = ....C3        FINI = .....C
*****************************************************
LABEL CYCLE    7                  TRUE LABELS                   CLOCK TIME =    7
                                /T(1)*P/
       A = ..1C34          T = .....2          C = ....C4        FINI = .....C
*****************************************************
LABEL CYCLE    8                  TRUE LABELS                   CLOCK TIME =    8
                                /T(2)*P/
       A = ..1C34          T = .....4          C = ....C4        FINI = .....C
*****************************************************
LABEL CYCLE    9                  TRUE LABELS                   CLOCK TIME =    9
                                /T(1)*P/
       A = ..1416          T = .....2          C = ....C5        FINI = .....C
*****************************************************
LABEL CYCLE   1C                  TRUE LABELS                   CLOCK TIME =   1C
                                /T(2)*P/
       A = ..1416          T = .....4          C = ....C5        FINI = .....C
*****************************************************
LABEL CYCLE   11                  TRUE LABELS                   CLOCK TIME =   11
                                /T(1)*P/
       A = ..16C7          T = .....2          C = ....C6        FINI = .....C
*****************************************************
LABEL CYCLE   12                  TRUE LABELS                   CLOCK TIME =   12
                                /T(2)*P/
       A = ..16C7          T = .....4
****************************************
LABEL CYCLE   13                  TRUE LAE
                                /T(1)*P/
       A = ..C7C3          T = .....2
*****************************************************
```

Fig. 10 Output of simulation of of the program in Fig. 9

```
$IBSYS
$*     MOUNT TAPE 1238 ON A9, RING OUT AND SAVE, THANK YOU
$PAUSE
$ATTACH          A9
$AS              SYSLB4
$REWIND          SYSLB4
$EXECUTE         USER
$ID    OUYANG   B.,*001/01/125*2M*100P*T$
$CDL2
$TRANSLATE       PRINT
*MAIN
COMMENT    ****MICRO-PROGRAMMED CONTR( L OF A
COMMENT        SERIAL COMPLEMENT SEQUENCE
C
 REGISTER,           A(12-1), C(0-3), H(0-2), F(0-10)
 REGISTER,           FINI
 MEMORY,             CM(H)=CM(0-7, 0-10)
 SWITCH,             START(ON)
 CLOCK,              P(2)
C
COMMENT    ****START THE COMPLEMENT SEQUENCE
 /START(ON)/         H=0, C=0, FINI=0, F=040
 /F( 3)*P(2)/        F=CM(H)
C
COMMENT    ****COMPLEMENT AND SHIFT
 /F( 7)*P(1)/        A(12-1)=A(1)'-A(12-2)
 /F( 6)*P(1)/        C=C.COUNT.
 /F( 5)*P(1)/        H=H.COUNT.
C/F( 3)*P(2)/        F=CM(H)
C
COMMENT    ****TEST AND BRANCH
 /F(10)*P(1)/        IF (C.EQ.12) THEN (H=H.COUNT.)
                      ELSE (H=F(0-2))
C/F( 3)*P(2)/        F=CM(H)
C
COMMENT    ****COMPLETION INDICATION
 /F( 4)*P(1)/        FINI=1
 END
$SIMULATE
*OUTPUT     CLOCK(1)=A,C,H,F,FINI
*SWITCH     1,START=ON
*LOAD
      A=707, CM(0-3)=56,41,20,0
*SIM        150,3
*RESET      CYCLE
'
$IBSYS
$RESTORE
```

Fig. 11 Listing of the description of a serial complement sequence in microprogram control

```
                 CUTPUT OF SIMULATICN

 SWITCH INTERRUPT
      START  = ON
        A = ..0707        C = ....00        H = .....0        F = ...040     FINI = .....0
 **************************************************
 LABEL CYCLE    1               TRUE LABELS               CLOCK TIME =    0
                               /F(3)*P(2)/
        A = ..0707        C = ....00        H = .....0        F = ...056     FINI = .....0
 **************************************************
 LABEL CYCLE    2               TRUE LABELS               CLOCK TIME =    1
                               /F(7)*P(1)/
                               /F(6)*P(1)/
                               /F(5)*P(1)/
        A = ..0343        C = ....01        H = .....1        F = ...056     FINI = .....0
 **************************************************
 LABEL CYCLE    3               TRUE LABELS               CLOCK TIME =    1
                               /F(3)*P(2)/
 **************************************************
 LABEL CYCLE    4               TRUE LABELS               CLOCK TIME =    2
                               /F(10)*P(1)/
        A = ..0343        C = ....01        H = .....0        F = ...041     FINI = .....0
 **************************************************
 LABEL CYCLE    5               TRUE LABELS               CLOCK TIME =    2
                               /F(3)*P(2)/
 **************************************************
 LABEL CYCLE    6               TRUE LABELS               CLOCK TIME =    3
                               /F(7)*P(1)/
                               /F(6)*P(1)/
                               /F(5)*P(1)/
        A = ..0161        C = ....02        H = .....1        F = ...056     FINI = .....0
 **************************************************
 LABEL CYCLE    7               TRUE LABELS               CLOCK TIME =    3
                               /F(3)*P(2)/
 **************************************************
 LABEL CYCLE    8               TRUE LABELS               CLOCK TIME =    4
                               /F(10)*P(1)/
        A = ..0161        C = ....02        H = .....0
 **************************************************
 LABEL CYCLE    9               TRUE LABELS
                               /F(3)*P(2)/
 **************************************************
 LABEL CYCLE   10               TRUE LABELS
                               /F(7)*P(1)/
                               /F(6)*P(1)/
                               /F(5)*P(1)/
        A = ..0070        C = ....03        H = .....1        F = ...056     FINI = .....0
```

Fig. 12, Output of simulation of the program in Fig. 11.

FORM 6413

```
C       *****   A SIMPLE MICROPROGRAMMED COMPUTER
C
 REGISTER,           R(0-21), A(0-21), C(0-13), D(0-13),
1                    F(1-22), H(1-5), G
 SUBREGISTER,        R(OP)=R(1-5), R(ADRS)=R(6-21), F(ADDR)=F(1-5)
 MEMORY,             M(C)=M(0-7777, 0-21), CM(H)=CM(0-13,1-22)
 SWITCH,             POWER(ON), START(ON), STOP(ON)
 CLOCK,              P(2)
C
COMMENT,   START COMPUTER OPERATION
C
 /POWER(ON)/         G=0, F=0, H=0, C=0, D=0
 /START(ON)/         G=1, F(10)=1
 /STOP(ON)/          G=0
C
COMMENT,   FETCH SEQUENCE WHEN H=0
C
 /F( 6)*P(0)/        R=M(C)
 /F( 7)*P(1)/        H=R(OP), C=R(ADRS), D=D.ADD.1
 /F(10)*P(2)/        IF (G) THEN (F=CM(H)) ELSE (H=0, C=0, D=0, R=0)
C
COMMENT,   ADD SEQUENCE WHEN H=1
C
C/F( 6)*P(0)/        R=M(C)
 /F(13)*P(1)/        A=A.ADD.R
 /F(11)*P(1)/        H=F(ADDR)
 /F(11)*P(2)/        C=D, F=CM(H)
C
COMMENT,   SUBTRACT SEQUENCE WHEN H=2
C
C/F( 6)*P(0)/        R=M(C)
 /F(14)*P(1)/        A=A.SUB.R
C/F(11)*P(1)/        H=F(ADDR)
C/F(11)*P(2)/        C=D, F=CM(H)
C
COMMENT,   JUMP-ON-MINUS SEQUENCE WHEN H=3
C
 /F(16)*P(0)/        IF (A(0)) THEN (D=R(ADRS))
C/F(11)*P(1)/        H=F(ADDR)
C/F(11)*P(2)/        C=D, F=CM(H)
C
COMMENT,   STORE SEQUENCE WHEN H=4
C
 /F(12)*P(0)/        R=A
 /F(12)*P(1)/        M(C)=R
C/F(11)*P(1)/        H=F(ADDR)
C/F(11)*P(2)/        C=D, F=CM(H)
C
COMMENT,   JUMP SEQUENCE WHEN H=5
C
 /F(15)*P(0)/        D=R(ADRS)
C/F(11)*P(1)/        H=F(ADDR)
C/F(11)*P(2)/        C=D, F=CM(H)
C
COMMENT,   SHIFT RIGHT ONE-BIT SEQUENCE WHEN H=6
C
 /F(17)*P(0)/        A(0-21)=0-A(0-20)
C/F(11)*P(1)/        H=F(ADDR)
C/F(11)*P(2)/        C=D, F=CM(H)
C
COMMENT,   CIRCULAR SHIFT LEFT ONE-BIT WHEN H=7
```

Fig. 13 Listing of the description of a microprogrammed computer

```
C*******
 /F(20)*P(0)/        A(0-21)=A(1-21)-A(0)
C/F(11)*P(1)/***    H=F(ADDR)**
C/F(11)*P(2)/        C=D, F=CM(H)
C
COMMENT,    CLEAR ADD SEQUENCE WHEN H=8
C
C/F( 6)*P(0)/        R=M(C)
 /F(21)*P(0)/        A=0
C/F(13)*P(1)/        A=A.ADD.R
C/F(11)*P(1)/        H=F(ADDR)
C/F(11)*P(2)/        C=D, F=CM(H)
C
COMMENT,    STOP SEQUENCE WHEN H=11
 /F(22)*P(0)/        G=0
C/F(11)*P(1)/        H=F(ADDR)
C/F(11)*P(2)/        C=D, F=CM(H)
 END
*SIMULATE
*OUTPUT    LABEL(1) =POWER,START,R,F,A,C,G,D,H,M(101),M(102),M(103)
*SWITCH    1,POWER=ON
*SWITCH    1,START=ON
*LOAD
 CM(0-7) =016000,011200,011100,001020,001400,001040,001010,001004,
 CM(10-11)=011202,001001,
 M(0-10)=100100,010101,040102,070000,030006,050003,040103,020102,060000,
   M(11)=110000, M(100-103)=000001,100001,000000,000000
*SIM       112,3
*RESET     CYCLE
```

Fig. 13   continued