AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH (AIUB)

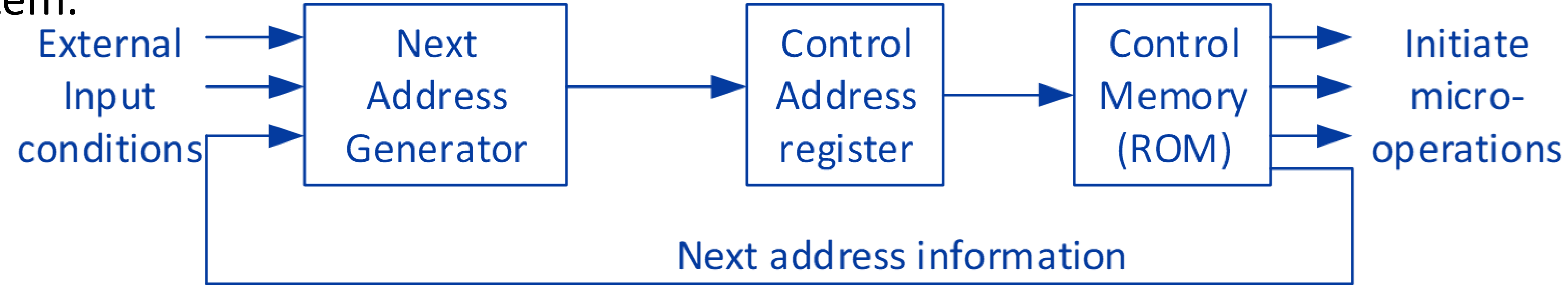
Where leaders are created

Lecture # 6 (Final)
Control Logic Design
Microprogram, flow-chart, state-diagram

Department of EEE

Microprogram Control

- In microprogram control, the control variables that initiate micro-operations are stored in memory.
- The control memory is usually a ROM (Read Only Memory).
- The control variables stored in memory are read once at a time to initiate the sequence of micro-operations for the system.
- The words stored in a control memory are micro-instructions, and each micro-instruction specifies one or more micro-operations for the components in the system.



Microprogram Control Contd..

- Inspection of state diagram reveals that the address sequencing in the microprogram control **must have** the following **capabilities**:
  - Provision for loading an external address as a result of the concurrence of external signals q_a and q_s.
  - Provision for sequencing consecutive addresses.
  - Provision for choosing between two addresses as a function of the present value of the status variables S and E.
- Each microinstruction must contain several bits to specify the way that the next address is to be selected.

AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Hardware Configuration

Table for the functions of the multiplexer select bits

ROM bits 13	14	MUX Select Function
0	0	Increment CAR
0	1	Load input to CAR
1	0	Load inputs to CAR if S = 1, increment CAR if S = 0
1	1	Load inputs to CAR if E = 1,increment CAR if E = 0

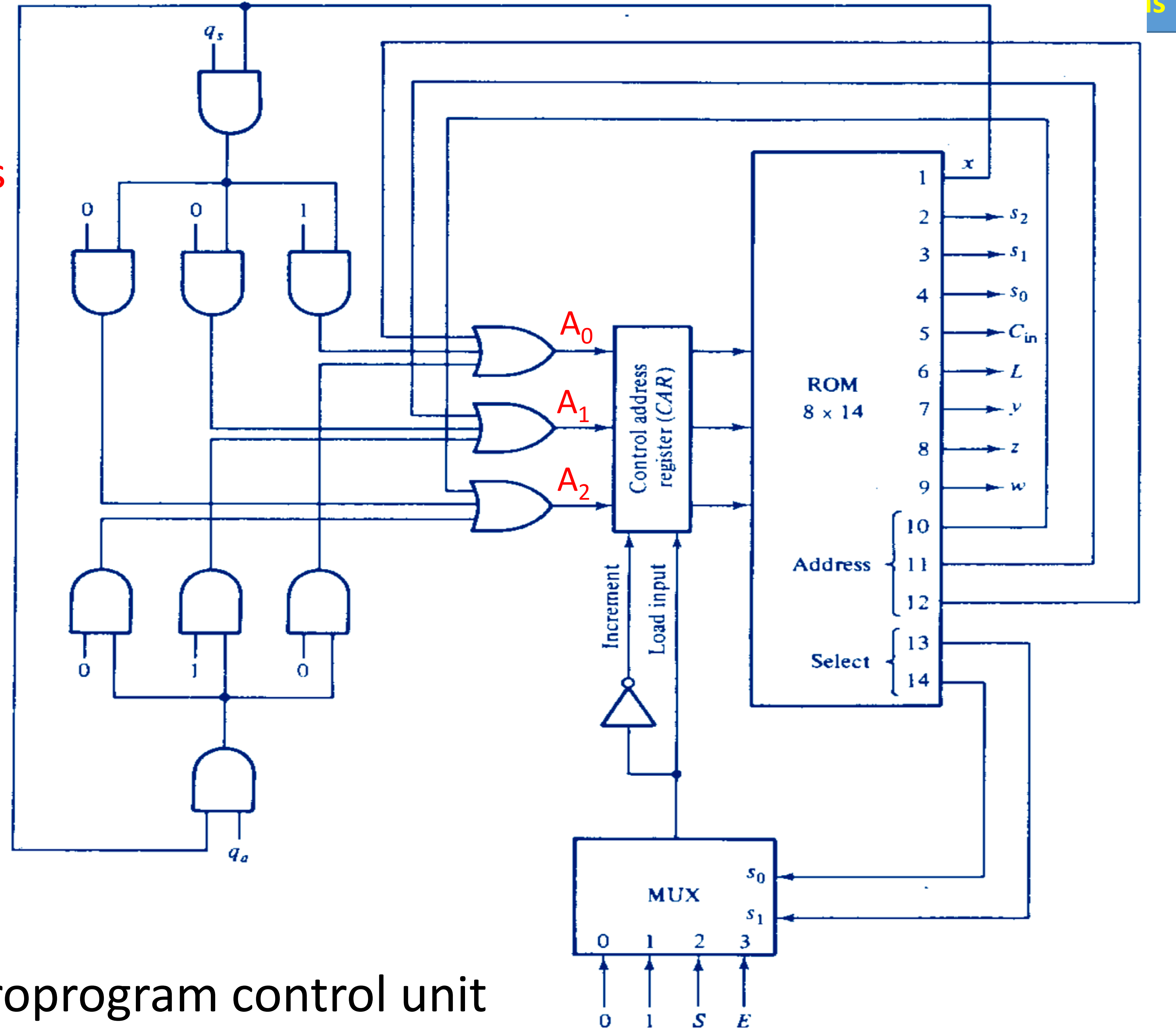


Fig. 10-10 Organization of the microprogram control unit

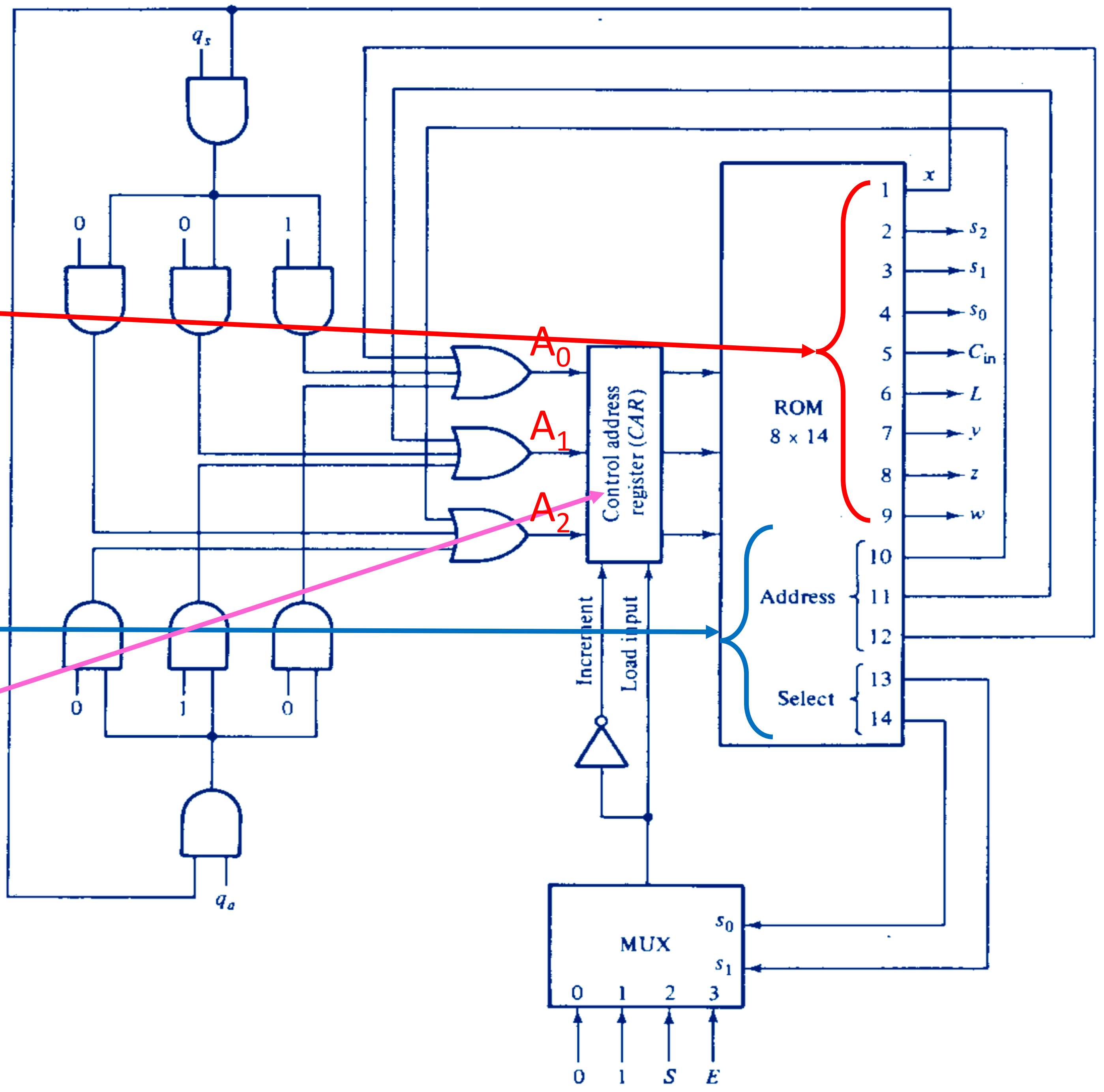
Hardware Configuration

The control memory is an 8-word (usually, 1 word = 2 contiguous 8-bit bytes, i.e., 1 word = 16 bits) by **14-bit ROM**.

The first 9 bits of a micro-instruction word contain the control variables that initiate the micro-operations.

The last 5 bits provide information to select the next address.

The Control Address Register (**CAR**) holds the address for the control memory. This register receives an input value when its load control is enabled; otherwise, it is incremented by 1. That is, CAR is a **counter** having parallel-load capability.



Hardware Configuration

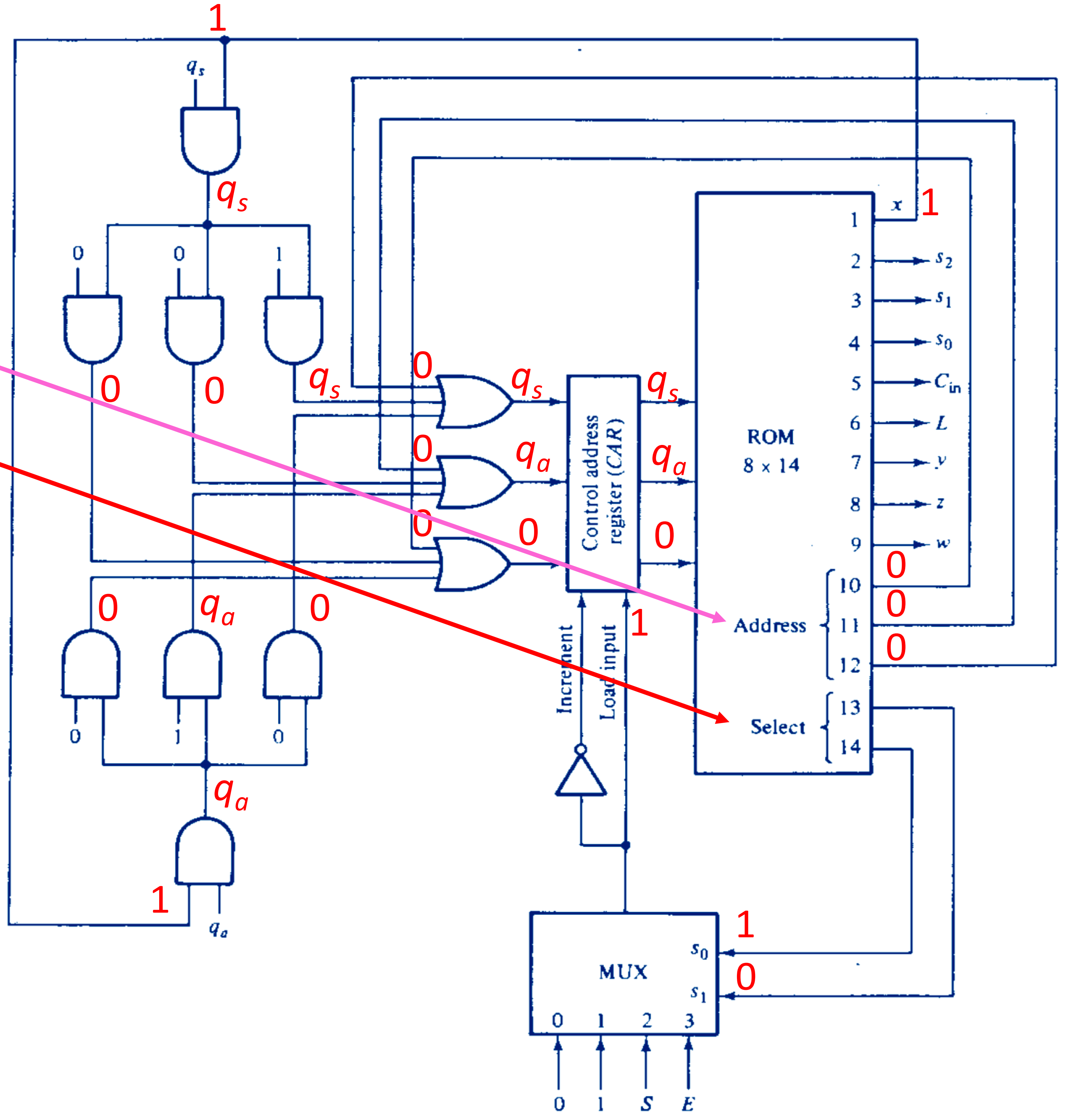
Bits 10, 11, and 12 of a micro-instruction contain an address for the CAR.

Bits 13 and 14 select an input for a multiplexer.

Bit 1 provides the initial state condition denoted by a variable *x* and also enables an external address when q_s and q_a are equal to 1.

We stipulate that when *x* = 1, the address field of the micro-instruction must be 000.

If both q_s and q_a are zero, addresses from bits 10, 11, and 12 are applied to the inputs of CAR. In this way, the control memory stays at address zero until an external variable is enabled.

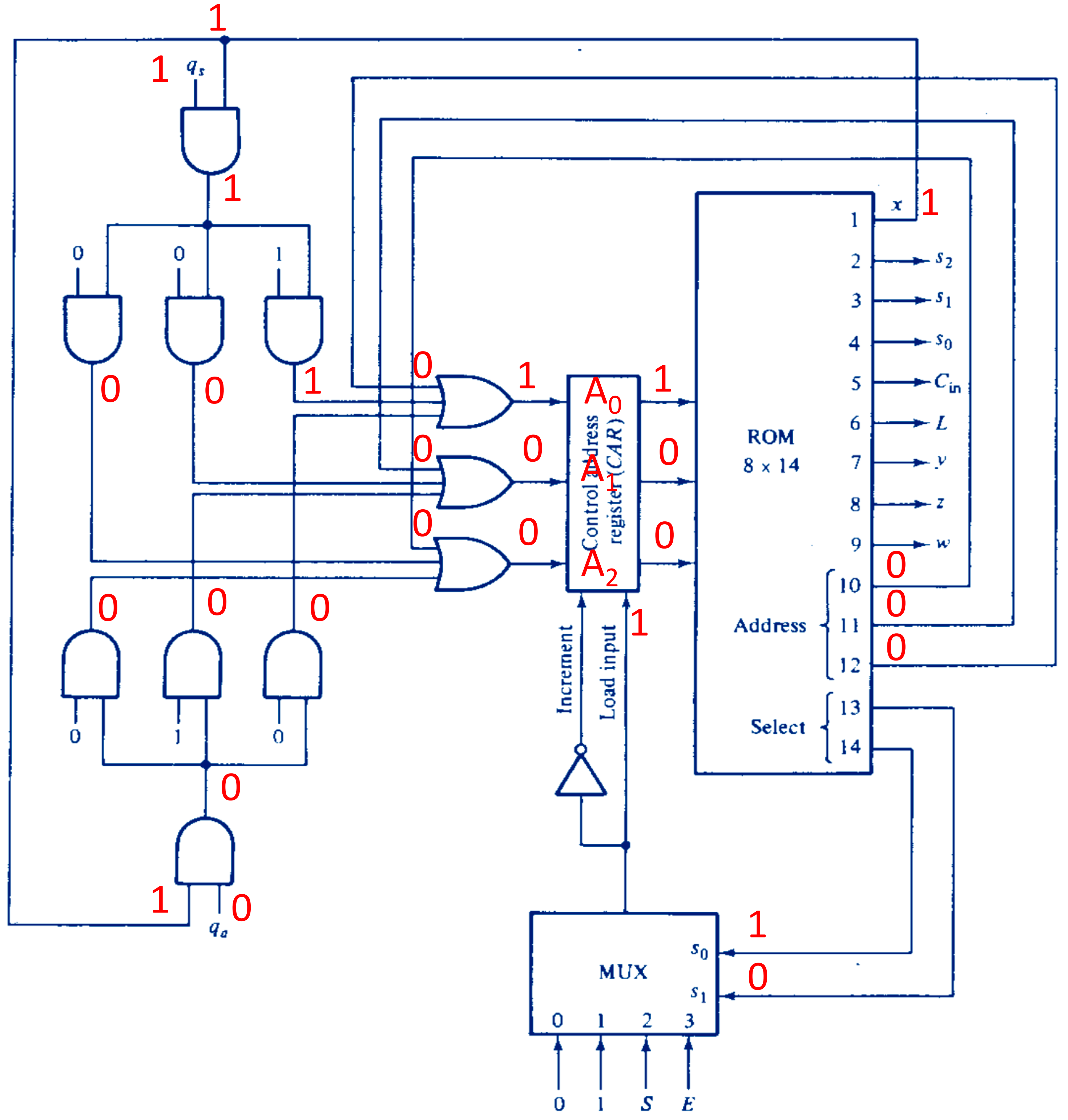


Hardware Configuration

If q_s = 1, address 001 is available at the inputs of the CAR.

Initially, x = 1.

An input of 1 is selected by the multiplexer when bits 13 and 14 are 01. The output of the multiplexer is 1, and the external input is loaded into CAR.

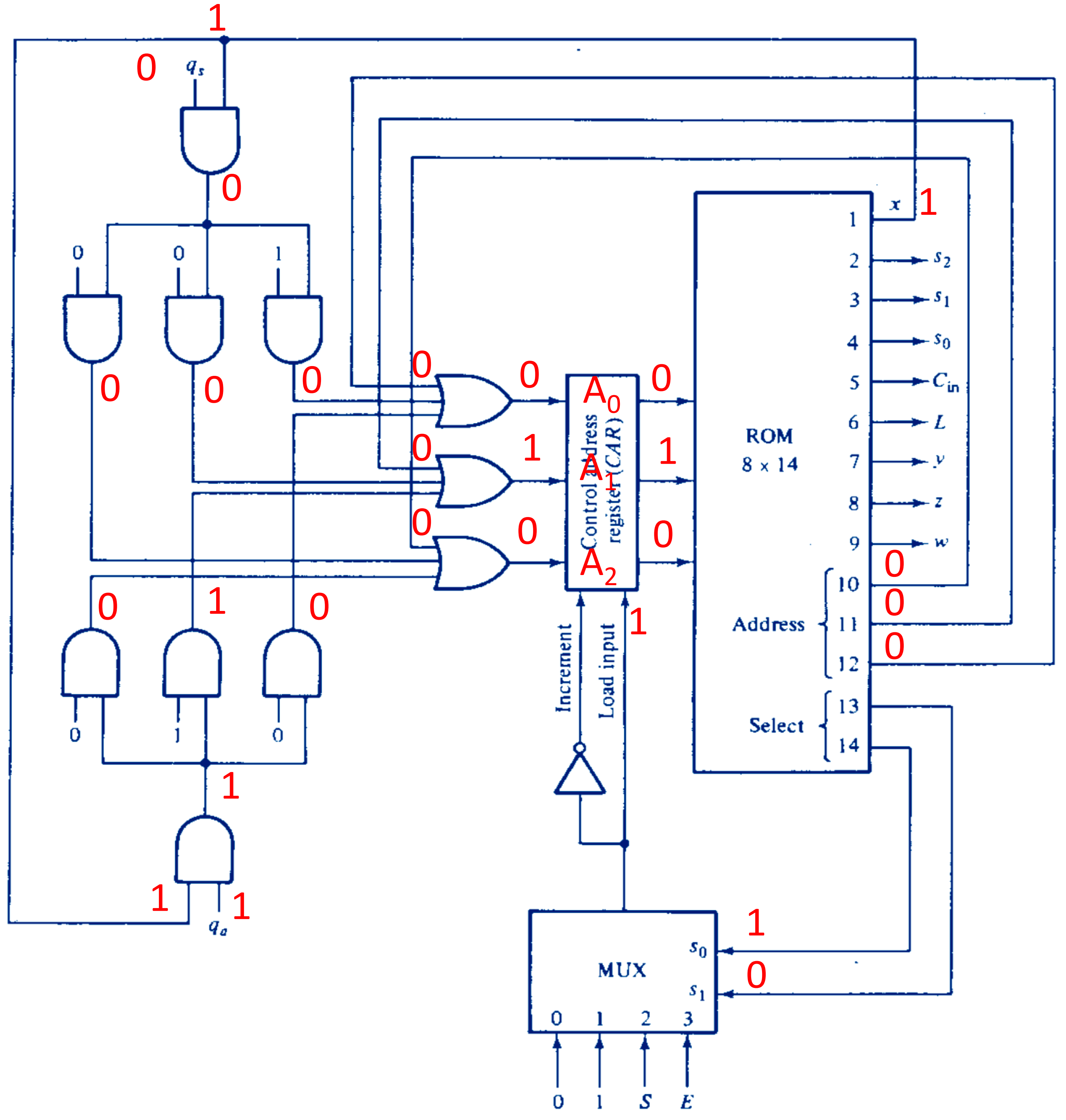


Hardware Configuration

If $q_a = 1$, address 010 is applied to the CAR.

Initially, $x = 1$.

An input of 1 is selected by the multiplexer when bits 13 and 14 are 01. The output of the multiplexer is 1, and the external input is loaded into CAR.



Hardware Configuration

The multiplexer (MUX) has four inputs that are selected with bits 13 and 14 of the micro-instructions. The functions of the multiplexer select bits are tabulated in the Table of Fig. 10-10.

If bits 13 and 14 are 00, a multiplexer input that is equal to 0 is selected. The output of the multiplexer is 0, and the increment input to CAR is enabled. This configuration increments CAR to choose the next address in sequence.

An input of 1 is selected by the multiplexer when bits 13 and 14 are 01. The output of the multiplexer is 1, and the external input is loaded into CAR.

The status variable *S* is selected when bits 13 and 14 are equal to 10. If $S = 1$, the output of the multiplexer is 1 and the address bits of the micro-instruction are loaded into the CAR (only if $x = 0$). If $S = 0$, the output of the multiplexer is 0 and the CAR is incremented.

AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Hardware Configuration

Table for the functions of the multiplexer select bits

ROM bits 13	14	MUX Select Function
0	0	Increment CAR
0	1	Load input to CAR
1	0	Load inputs to CAR if S = 1, increment CAR if S = 0
1	1	Load inputs to CAR if E = 1,increment CAR if E = 0

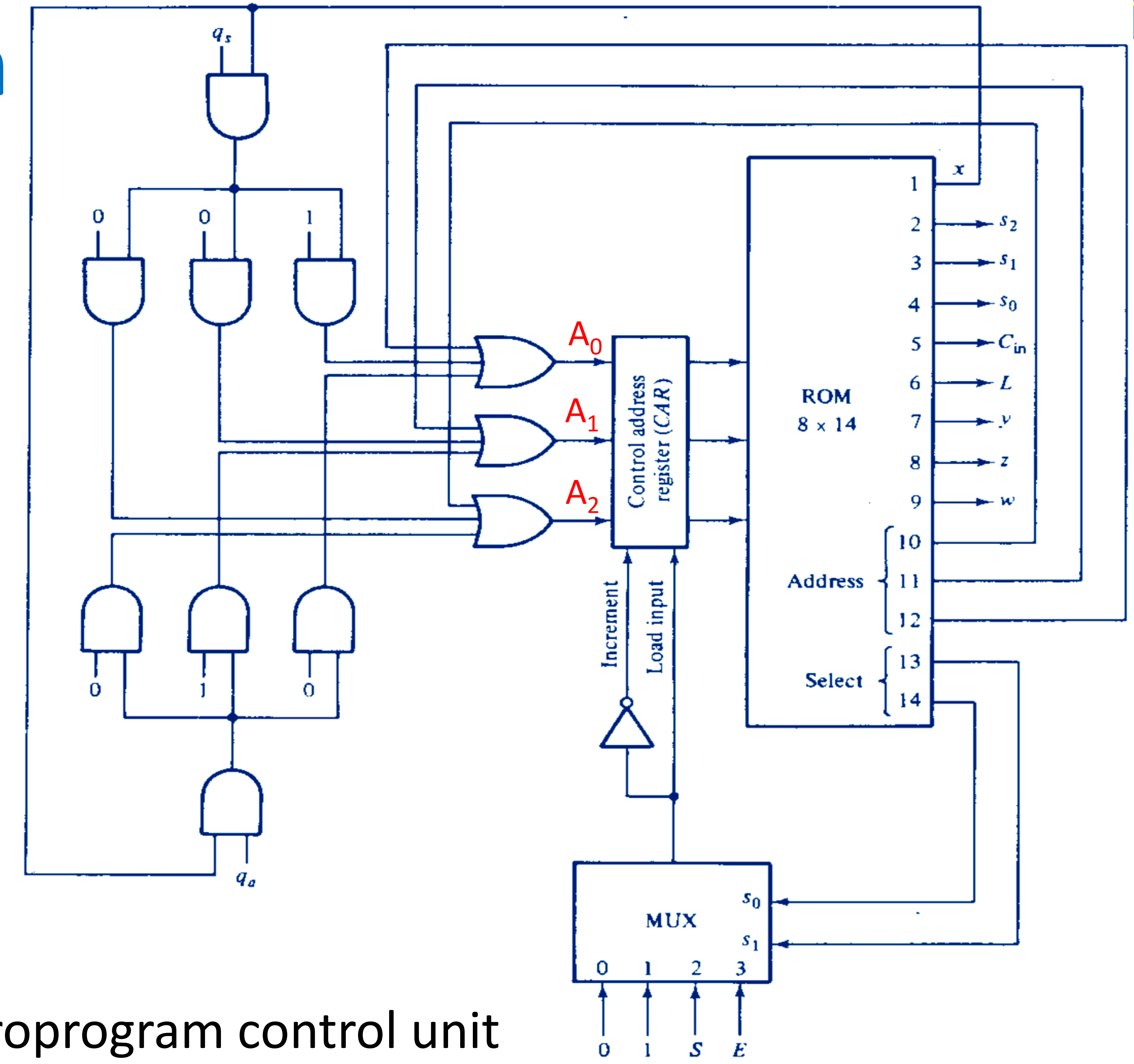


Fig. 10-10 Organization of the microprogram control unit

Hardware Configuration

With bits 13 and 14 equal to 11, the status variable *E* is selected, and the address bits of the micro-instruction are loaded into the CAR if *E* = 1, but the CAR is incremented if *E* = 0. Thus, the multiplexer allows the control to choose between two addresses, depending on the value of the selected status bit.

Once the configuration of a micro-program control unit is established, the designer's task is to generate the micro-code for the control memory. This code generation is called micro-programming and is a process that determines the bit configuration for each all words in control memory.

To appreciate this process, we will derive the micro-program for the adder-subtractor example. The control memory has 8 words, and each word contains 14 bits. To micro-program the control memory, we must determine the bit values of each of the eight words.

Hardware Configuration

The register-transfer method can be adopted for developing a micro-program. The micro-program sequence can be specified with register-transfer statements. There is no need for listing control functions with Boolean variables since, in this case, the control variables are the control words stored in control memory. Instead of a control function, we specify an address with each register-transfer statement.

The address associated with each symbolic statement corresponds to the address where the micro-instruction is to be stored in memory.

The sequencing from one address to the next can be indicated by means of conditional control statements. This type of statement can specify the address to which control goes, depending on status conditions.

Hardware Configuration

Thus, instead of thinking in terms of the 1's and 0's that must be inserted for each micro-instruction, it is more convenient to think in terms of symbols in the register-transfer method. Once the symbolic micro-program is established, it is possible to translate the register-transfer statements to their equivalent binary form.

The micro-program in symbolic form is given in Table 10-2. The eight addresses of the ROM are listed in the first column.

In the second column, the micro-instruction that must be stored at each address is given in symbolic form. The comments are used to clarify the register-transfer statements.

AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Microprogram for Control Memory

Table 10-2 Microprogram for Control Memory

ROM address	Microinstruction	Comments
0	$x = 1$, if $(q_s = 1)$ then (go to 1), if $(q_a = 1)$ then (go to 2), if $(q_s \wedge q_a = 0)$ then (go to 0)	Load 0 or external address
1	$B_s \leftarrow \bar{B}_s$	$q_s = 1$, start subtraction
2	If $(S = 1)$ then (go to 4)	$q_a = 1$, start addition
3	$A \leftarrow A + B$, $E \leftarrow C_{out}$, go to 0	Add magnitudes and return
4	$A \leftarrow A + \bar{B} + 1$, $E \leftarrow C_{out}$,	Subtract magnitudes
5	If $(E = 1)$ then (go to 0), $E \leftarrow 0$	Operation terminated if $E = 1$
6	$A \leftarrow \bar{A}$	$E = 0$, complement A
7	$A \leftarrow A + 1$, $A_s \leftarrow \bar{A}_s$, go to 0	Done, return to address 0

Control State Diagram

T_0: Initial state $x = 1$

$T_1 : B_s \leftarrow \bar{B}_s$

T_2: nothing

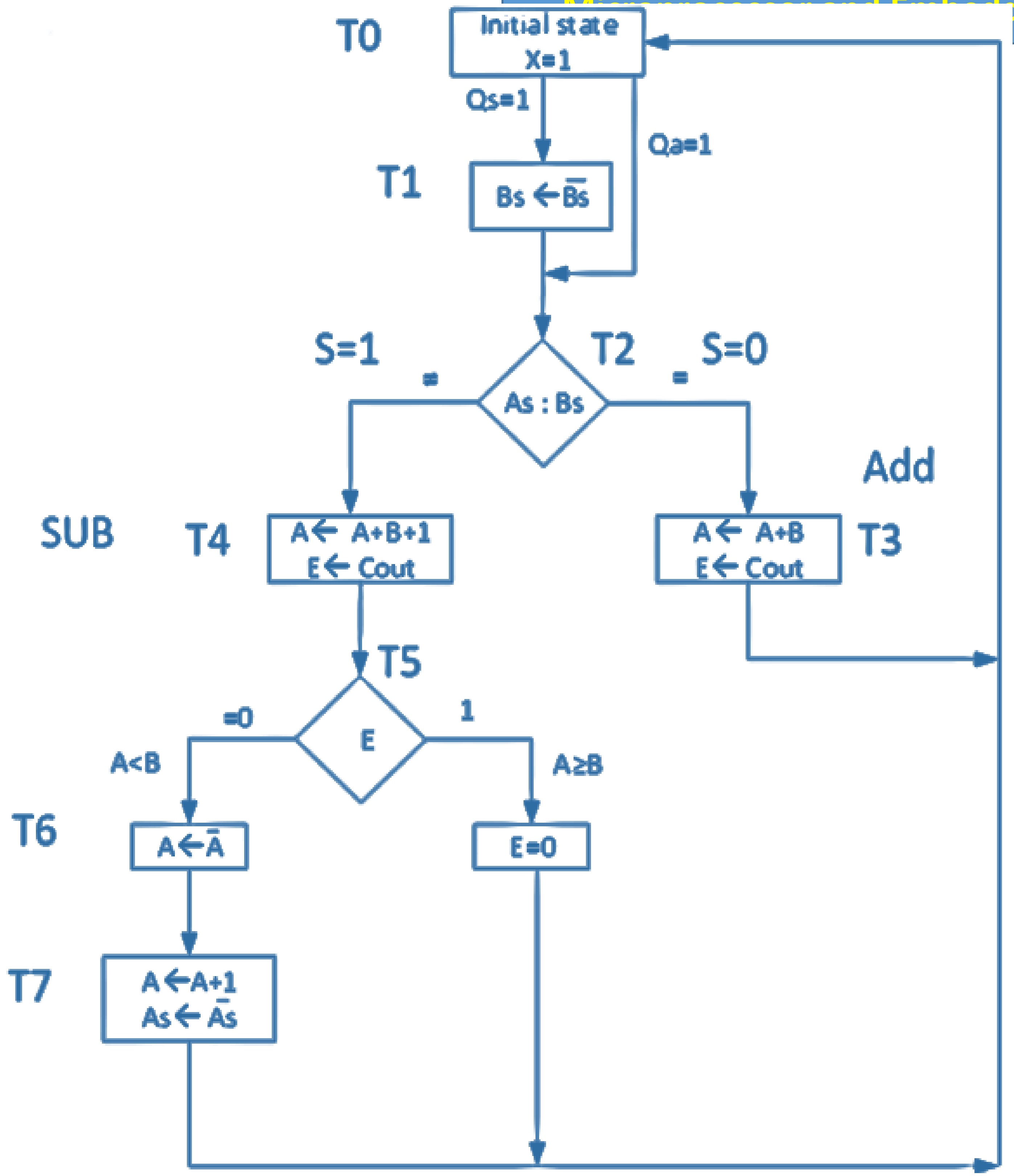
$T_3 : A \leftarrow A + B,\ E \leftarrow C_{out}$

$T_4 : A \leftarrow A + \bar{B} + 1,\ E \leftarrow C_{out}$

$T_5 : E \leftarrow 0$

$T_6 : A \leftarrow \bar{A}$

$T_7 : A \leftarrow A + 1,\ A_s \leftarrow \bar{A}_s$



Hardware Configuration

The equivalent binary form of micro-program is given in Table 10-3. The eight binary format addresses of the ROM are listed in the first column.

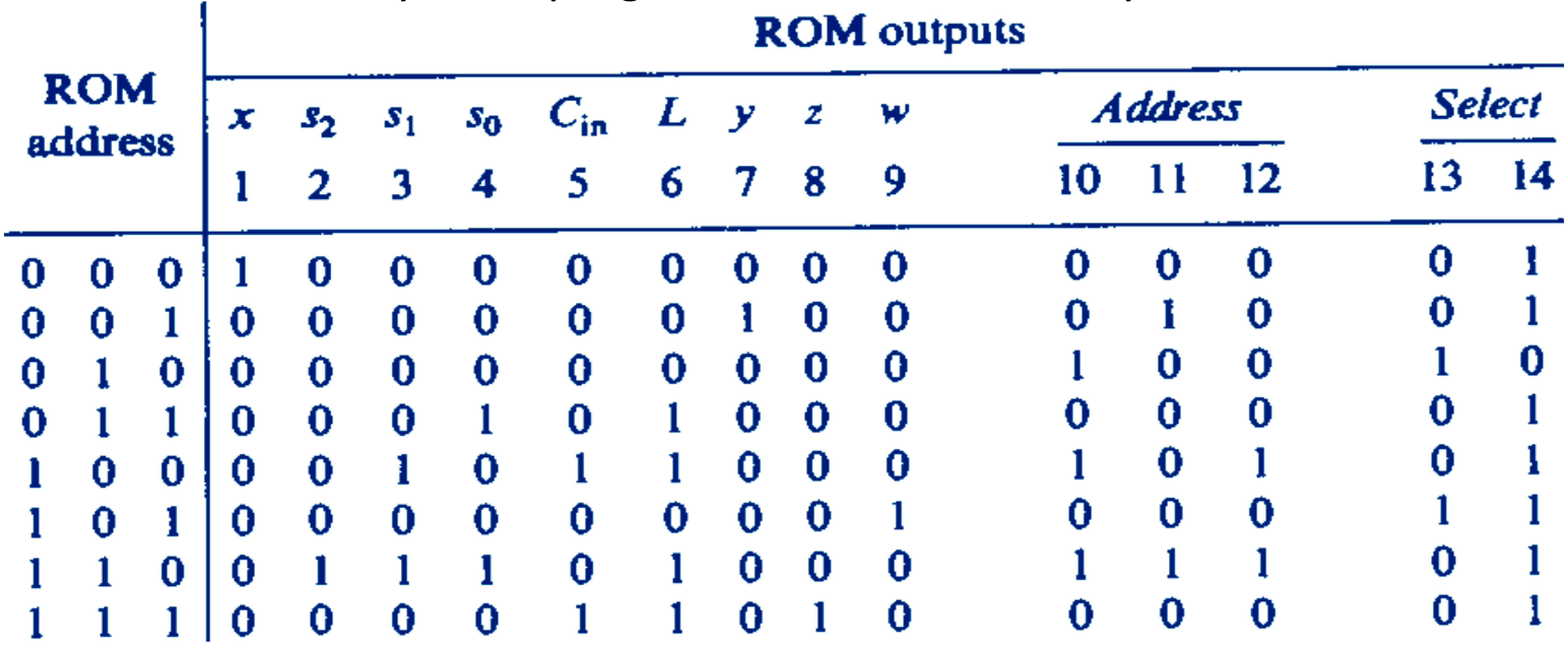
In the second column, the micro-instruction that must be stored at each address is given in the binary or machine language format.

In the third column, the contents of each word of ROM is given in binary format. This is for programming the ROM.

The first 9 bits in each ROM word give the control word that initiates the specified micro-operations. These are taken from Fig. **10-9 (b).** The last 5 bits in each ROM word are derived from the conditional control statements in the symbolic program.

Microprogram for Control Memory

Table 10-3 Binary microprogram for control memory



ROM address			ROM outputs													
			x	s_2	s_1	s_0	C_{in}	L	y	z	w	Address			Select	
			1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	1
0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
0	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0	1
1	0	0	0	0	1	0	1	1	0	0	0	1	0	1	0	1
1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1
1	1	0	0	1	1	1	0	1	0	0	0	1	1	1	0	1
1	1	1	0	0	0	0	1	1	0	1	0	0	0	0	0	1

Hardware Configuration

Address 0 is equivalent to the initial state and produces an output $x = 1$. The next address depends on the values of external variables q_s and q_a.

The three conditional control statements in this micro-instruction use a *go to* statement, control goes to the address written after the words *go to*.

Thus, if both q_s and q_a are 0, control stays in address 0 to repeat the micro-instruction.

If q_s or q_a is 1, control goes to address 1 or 2, respectively.

Hardware Configuration

Table for the functions of the multiplexer select bits

ROM bits 13	ROM bits 14	MUX Select Function
0	0	Increment CAR
0	1	Load input to CAR
1	0	Load inputs to CAR if S = 1, increment CAR if S = 0
1	1	Load inputs to CAR if E = 1,increment CAR if E = 0

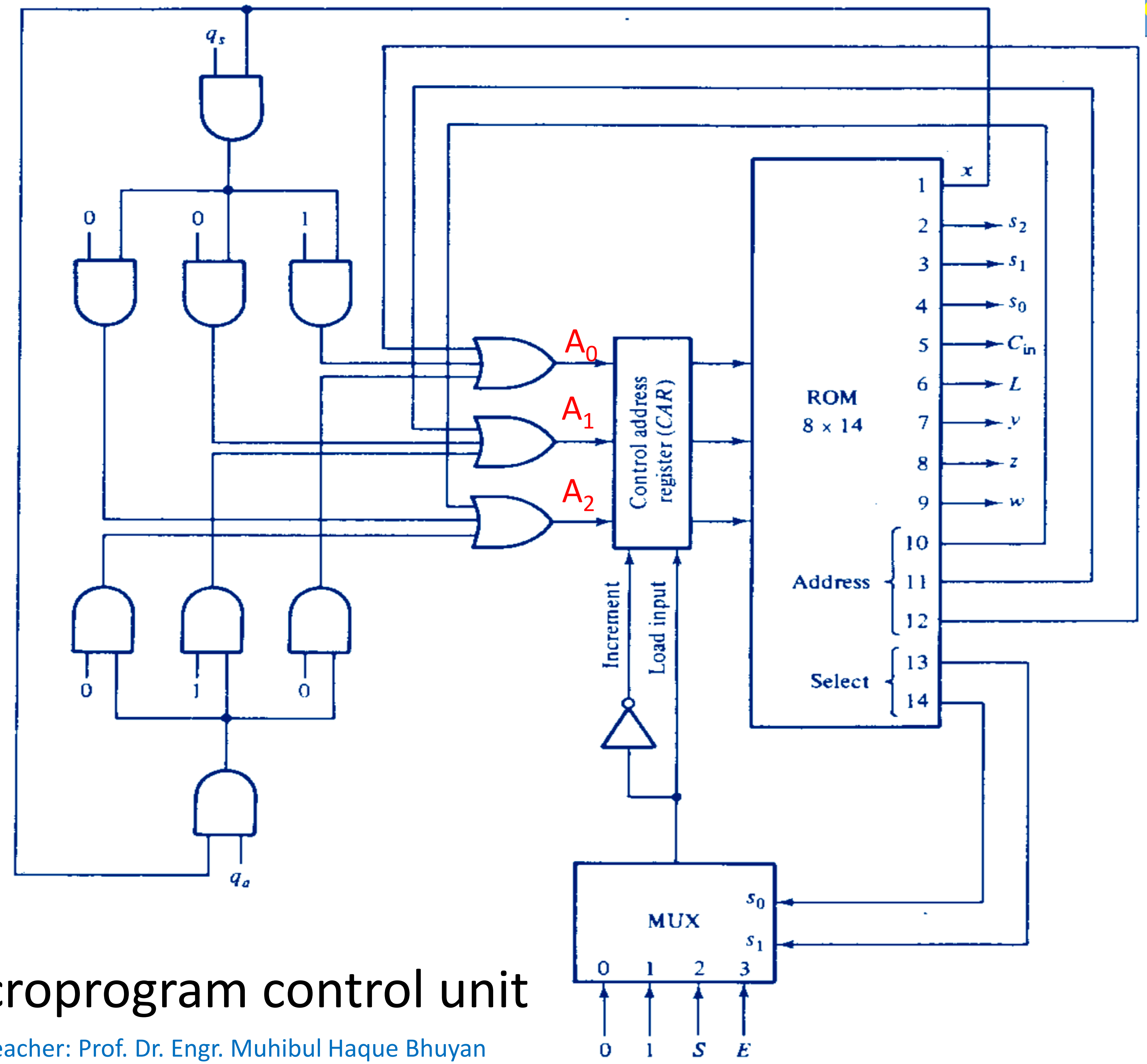


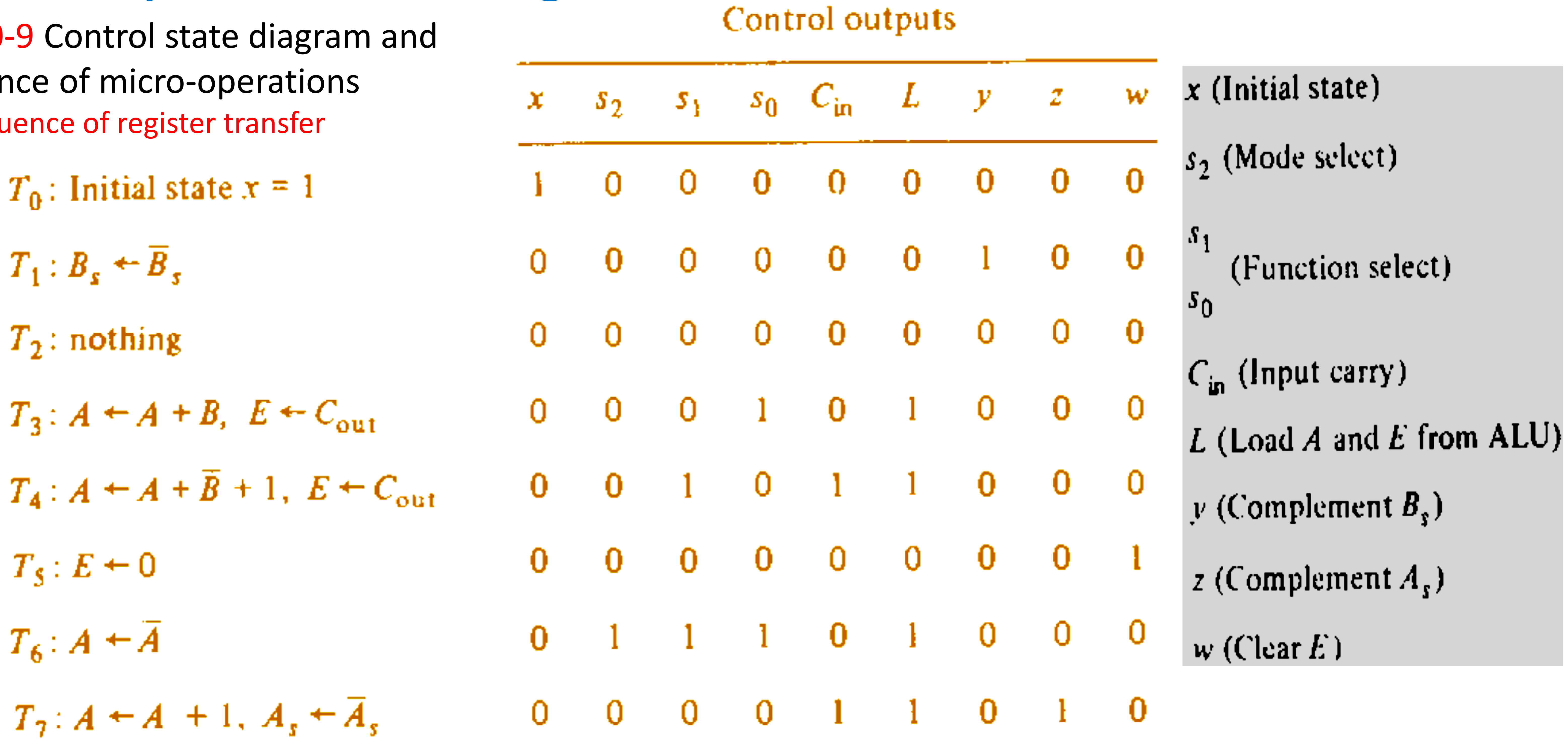
Fig. 10-10 Organization of the microprogram control unit

AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Sequence of Register Transfers

Fig. 10-9 Control state diagram and sequence of micro-operations

(b) Sequence of register transfer



	Control outputs								
	x	s_2	s_1	s_0	C_{in}	L	y	z	w
T_0: Initial state $x = 1$	1	0	0	0	0	0	0	0	0
$T_1: B_s \leftarrow \overline{B}_s$	0	0	0	0	0	0	1	0	0
T_2: nothing	0	0	0	0	0	0	0	0	0
$T_3: A \leftarrow A + B,\ E \leftarrow C_{out}$	0	0	0	1	0	1	0	0	0
$T_4: A \leftarrow A + \overline{B} + 1,\ E \leftarrow C_{out}$	0	0	1	0	1	1	0	0	0
$T_5: E \leftarrow 0$	0	0	0	0	0	0	0	0	1
$T_6: A \leftarrow \overline{A}$	0	1	1	1	0	1	0	0	0
$T_7: A \leftarrow A + 1,\ A_s \leftarrow \overline{A}_s$	0	0	0	0	1	1	0	1	0

x (Initial state)

s_2 (Mode select)

s_1, s_0 (Function select)

C_{in} (Input carry)

L (Load A and E from ALU)

y (Complement B_s)

z (Complement A_s)

w (Clear E)

Hardware Configuration

The conditional control statements in the other micro-instructions use the status variables *S* and *E*. The *go to* statement without a condition attached specifies an unconditional branch to the indicated address.

For example, *go to* 0 means that control goes to address 0 after the present micro-instruction is executed. If there is no *go to* statement in the micro-instruction, it implies that the next micro-instruction is taken from the next address in sequence. Also, if the condition after an *if* statement is not satisfied, control goes to the next address in sequence.

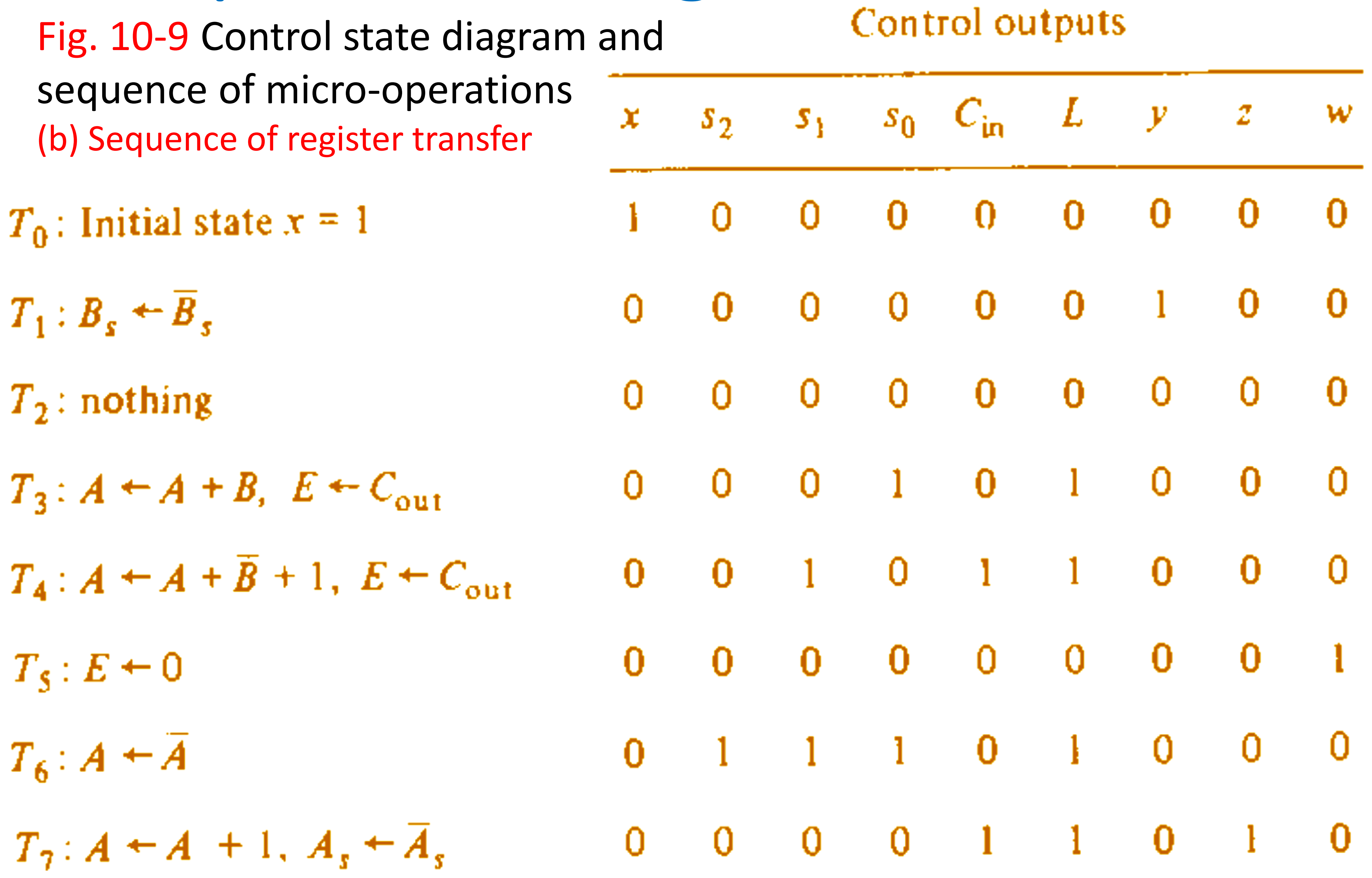
Hardware Configuration

The micro-instructions associated with the eight addresses are derived directly from the control specifications of Fig. 10-9. The micro-instructions listed are identical to the ones listed in Fig. 10-9 (b). The conditional control statement specifies the address sequence as given by the state diagram of Fig. 10-9 (a).

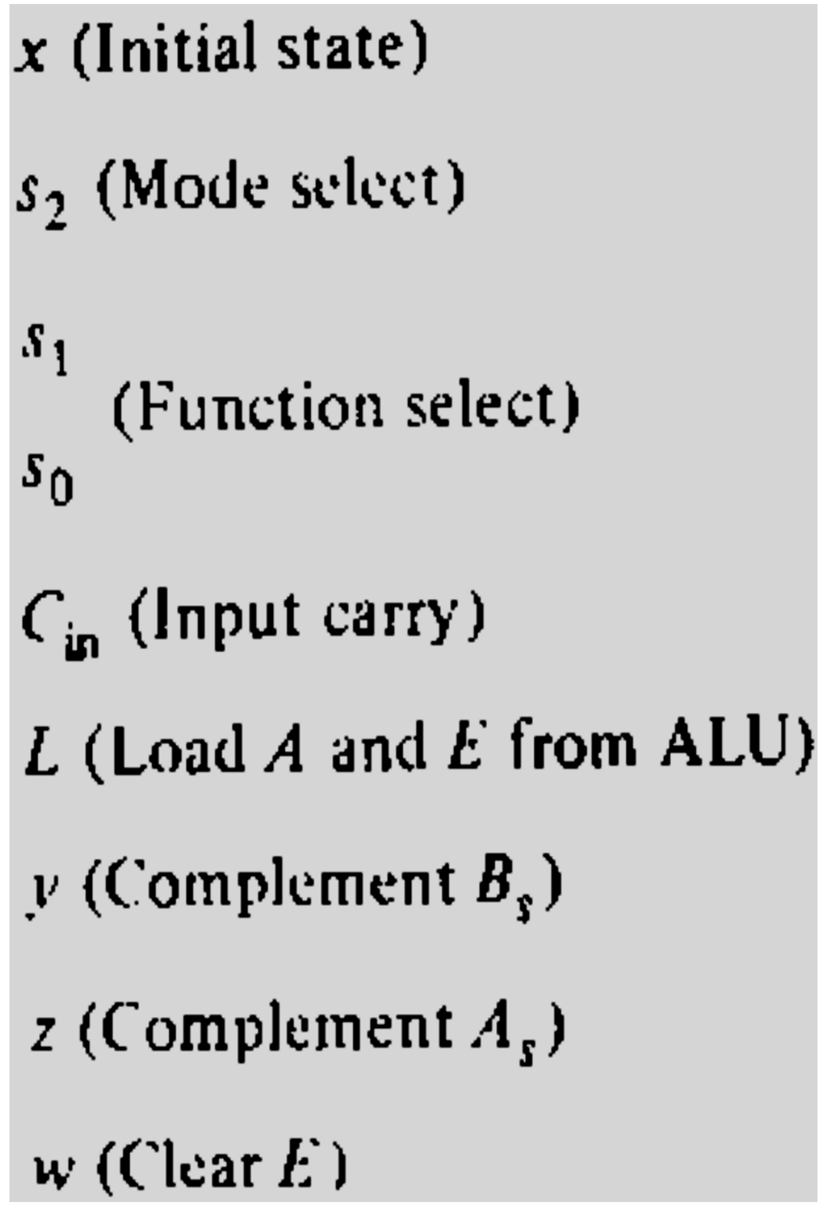
Note that each address number is the same as the subscript number under the *T*'s in the state diagram. It should be obvious that the conditional control statements provide a different way to specify a state diagram. This shows that the register-transfer method can be used to specify a sequential circuit.

Sequence of Register Transfers

Fig. 10-9 Control state diagram and sequence of micro-operations
(b) Sequence of register transfer



	Control outputs								
	x	s_2	s_1	s_0	C_{in}	L	y	z	w
T_0: Initial state $x = 1$	1	0	0	0	0	0	0	0	0
$T_1: B_s \leftarrow \overline{B}_s$	0	0	0	0	0	0	1	0	0
T_2: nothing	0	0	0	0	0	0	0	0	0
$T_3: A \leftarrow A + B,\ E \leftarrow C_{out}$	0	0	0	1	0	1	0	0	0
$T_4: A \leftarrow A + \overline{B} + 1,\ E \leftarrow C_{out}$	0	0	1	0	1	1	0	0	0
$T_5: E \leftarrow 0$	0	0	0	0	0	0	0	0	1
$T_6: A \leftarrow \overline{A}$	0	1	1	1	0	1	0	0	0
$T_7: A \leftarrow A + 1,\ A_s \leftarrow \overline{A}_s$	0	0	0	0	1	1	0	1	0



x	(Initial state)
s_2	(Mode select)
s_1, s_0	(Function select)
C_{in}	(Input carry)
L	(Load A and E from ALU)
y	(Complement B_s)
z	(Complement A_s)
w	(Clear E)

Control State Diagram

T_0 : Initial state $x = 1$

$T_1 : B_s \leftarrow \bar{B}_s$

T_2 : nothing

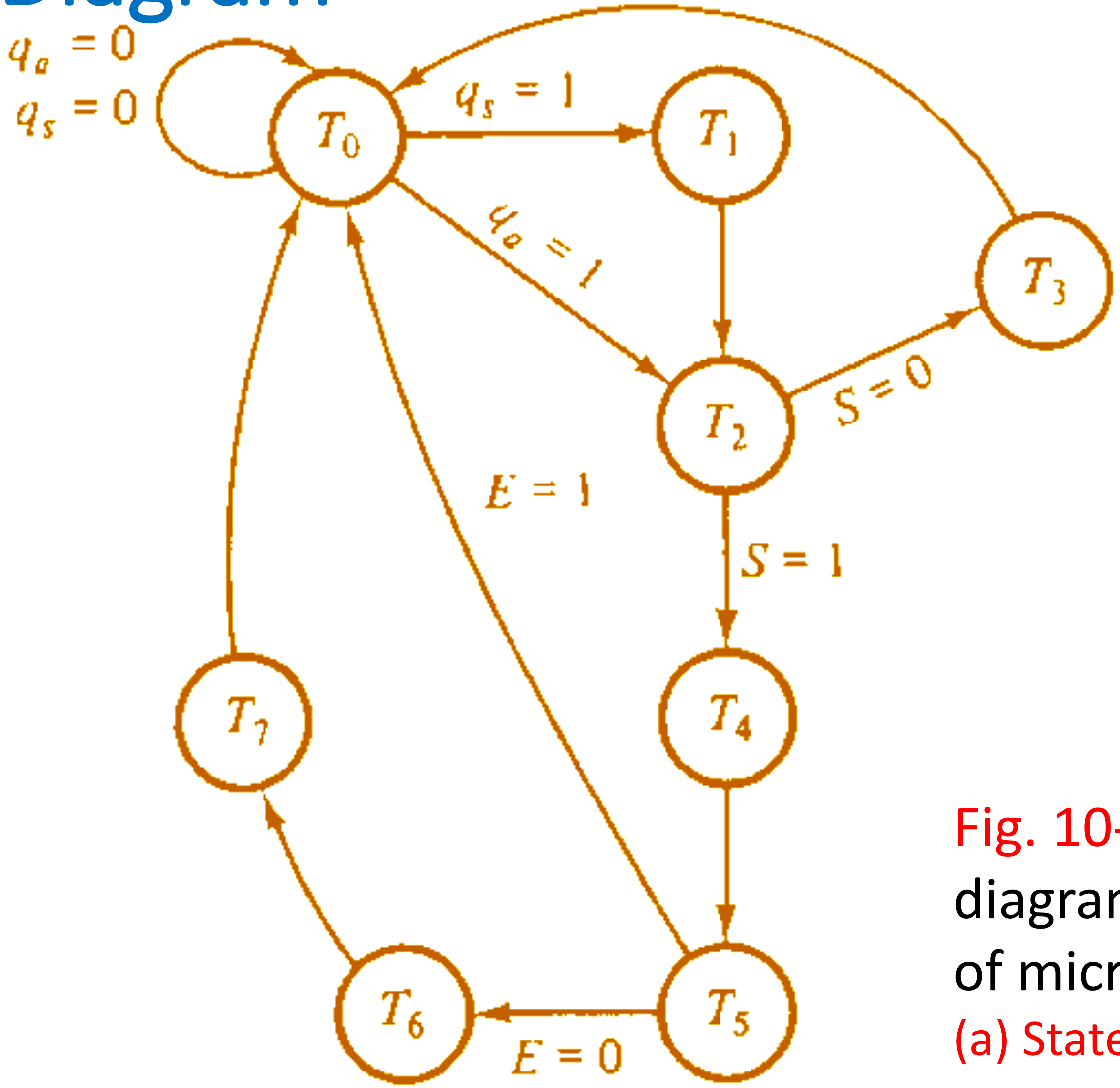
$T_3 : A \leftarrow A + B,\ E \leftarrow C_{out}$

$T_4 : A \leftarrow A + \bar{B} + 1,\ E \leftarrow C_{out}$

$T_5 : E \leftarrow 0$

$T_6 : A \leftarrow \bar{A}$

$T_7 : A \leftarrow A + 1,\ A_s \leftarrow \bar{A}_s$



q_a	Add
q_s	Subtract
$S = 0$	Signs alike
$S = 1$	Signs unlike
E	Output carry

Fig. 10-9 Control state diagram and sequence of micro-operations
(a) State diagram

Control of Processor Unit

- To construct correct micro-programs, it is necessary to specify exactly how the status bits are affected by each micro-operation in the processor.
- The S (sign) and Z (Zero) bits are affected by all micro-operations.
- The C (Carry) and V (Overflow) bits do not change after the following ALU operations:
  - The four logic operations OR, AND, XOR, and NOT.
  - The increment and decrement operations.

Control of Processor Unit

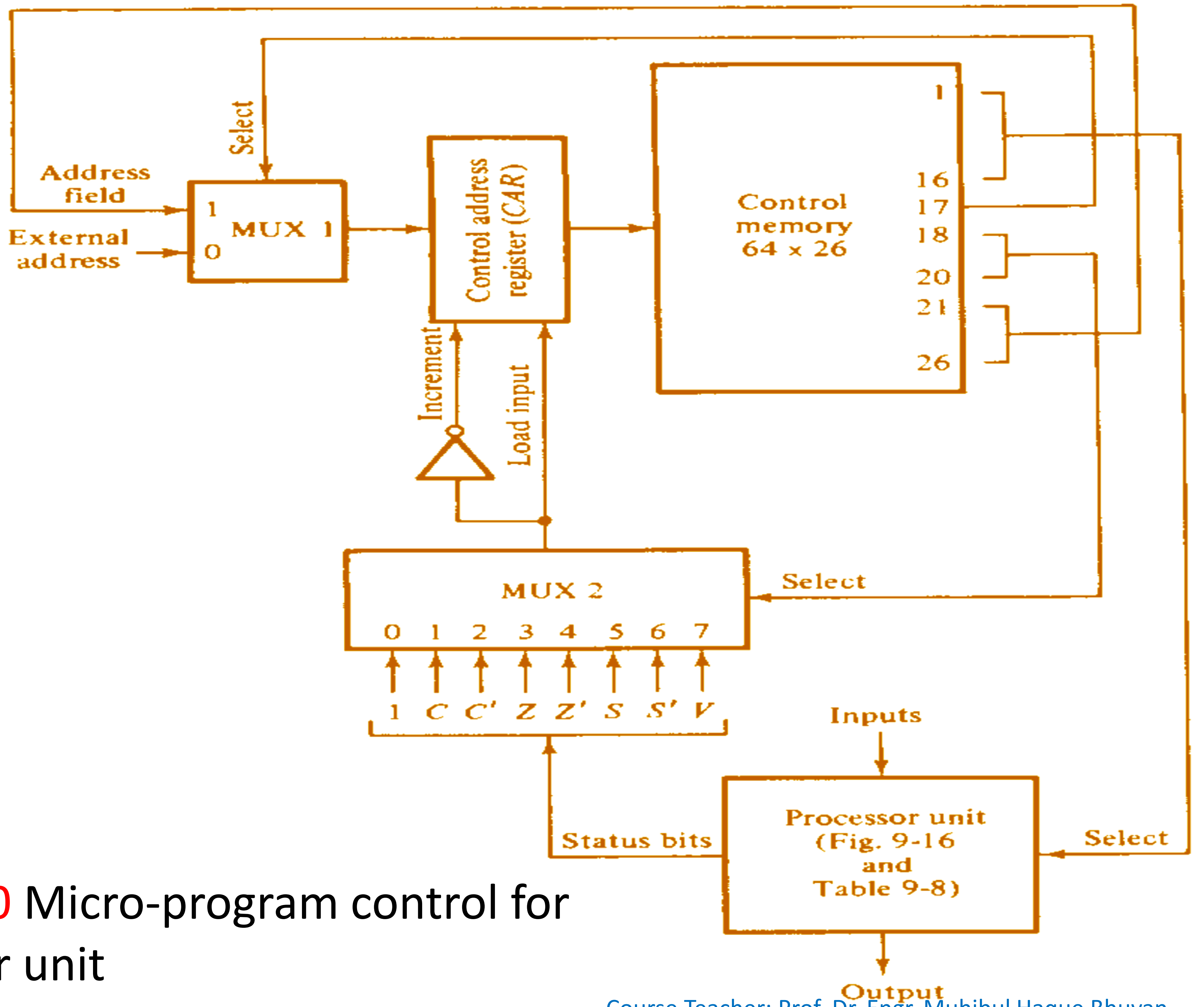
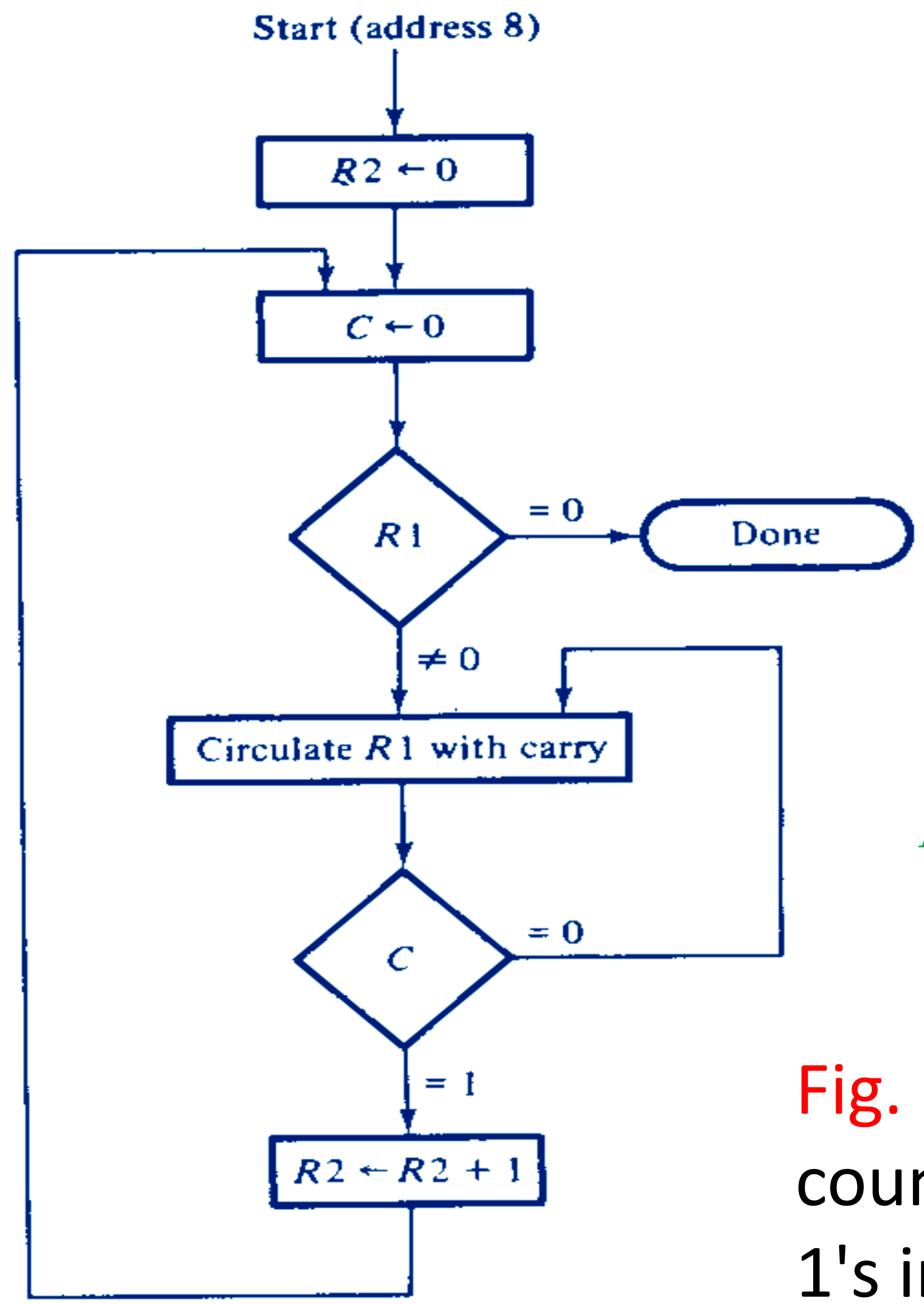


Fig. 10-10 Micro-program control for processor unit

Example of Micro-program

- We will now demonstrate by means of an example how a micro-program is written to implement a given macro-operation.
- A macro-operation initiates a sequence of micro-instructions in the control memory.
- This sequence constitutes a micro-program routine for executing the specified macro-operation.
- A macro-operation is initiated by an external address that supplies the first address in the control memory for the micro-instruction routine.
- The routine is terminated with a micro-instruction that loads a new external address to start executing the next macro-operation.

Flow chart for counting the number of 1's in register R1



The macro-operation, we wish to implement, counts the number of 1's presently stored in the processor register, R1, and sets the register, R2 to that number.
For example, if R1 = 00110101, the micro-program routine counts that there are four 1s stored in the R1 register and thus it sets the R2 register value to 4, i.e., 00000100b.

Fig. 10-12 Flowchart for counting the number of 1's in register R1

A zero (0) is inserted at the MSB →

R1 (Binary)								C	R2 (D)
0	0	1	1	0	1	0	1	0	0
0	0	0	1	1	0	1	0	1	1
0	0	0	0	1	1	0	1	0	1
0	0	0	0	0	1	1	0	1	2
0	0	0	0	0	0	1	1	0	2
0	0	0	0	0	0	0	1	1	3
0	0	0	0	0	0	0	0	1	4

Binary microprogram to count the number of 1’s in R1

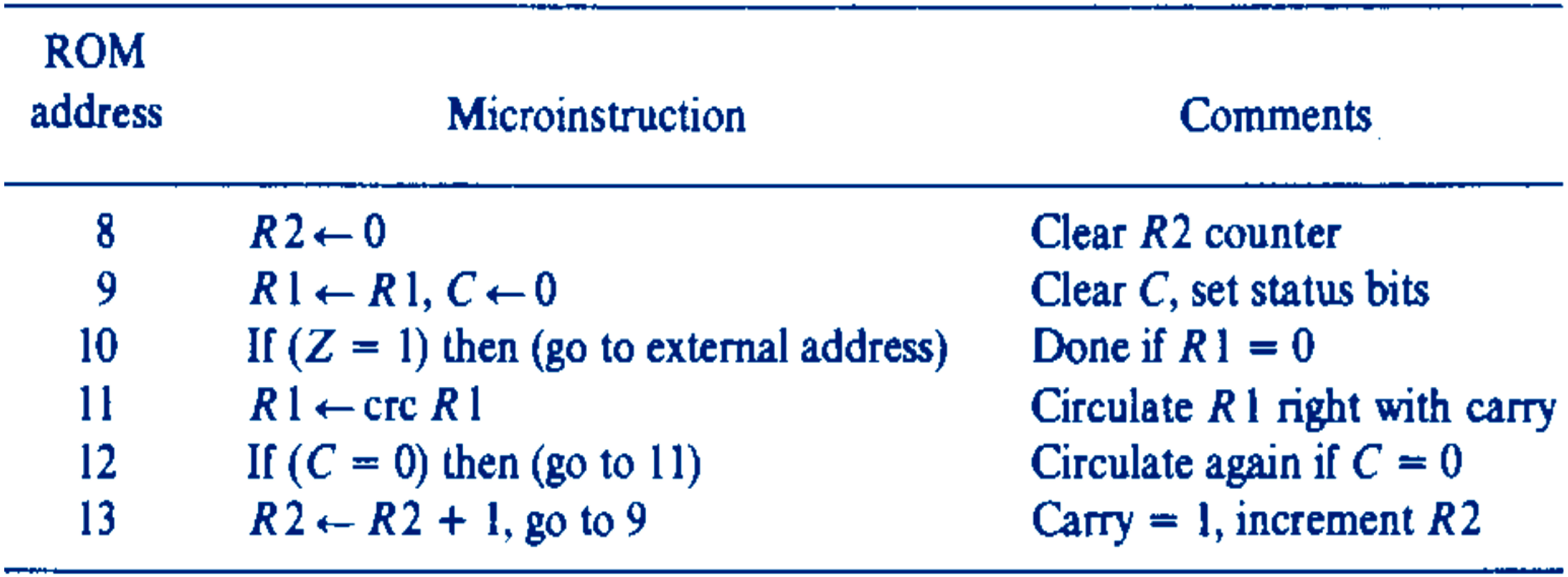
- Flowchart shows the sequence of micro-operations and decision paths.
- We assume that the micro-program routine starts at address 8.
- Register R2 and the carry bit (C) are first set to 0.
- The contents of R1 is then examined.
  - If it is 0, it signifies that there is no 1's stored in it, so the micro-program routine terminates with R2 = 0.
  - If the contents of R1 is not 0, it indicates that there are some 1's stored in it.
- Register, R1 together with the carry is shifted (to either left or right side) in a circular manner as many times as necessary until a 1 is transferred into C.

Binary microprogram to count the number of 1’s in R1

- For every 1 detected in C, we increment register R2 and then go back to check if R1 = 0.
- This loop is repeated until all the 1's in R1 are counted. The value of C is always 0 when it is circulated with the contents of R1.
- The micro-program routine in symbolic form is given in Table 10-4.
- The routine starts at address 8 by clearing register, R2.
- The micro-instruction in address 9 clears the C bit and sets the Z bit if R1 contains all 0's. This is done by transferring the contents of R1 into itself through the ALU.

Binary microprogram to count the number of 1's in R1

Table 10-4 Symbolic micro-program



ROM address	Microinstruction	Comments
8	$R2 \leftarrow 0$	Clear $R2$ counter
9	$R1 \leftarrow R1, C \leftarrow 0$	Clear C, set status bits
10	If $(Z = 1)$ then (go to external address)	Done if $R1 = 0$
11	$R1 \leftarrow$ crc $R1$	Circulate $R1$ right with carry
12	If $(C = 0)$ then (go to 11)	Circulate again if $C = 0$
13	$R2 \leftarrow R2 + 1$, go to 9	Carry = 1, increment $R2$

Binary microprogram to count the number of 1's in R1

- The micro-instruction in address 10 checks the value of the Z bit. If it is 1, it indicates that R1 contains all 0's, and the routine is terminated by accepting a new external address to start executing another macro-operation.
- If Z is not equal to 1, control continues with address 11.
- The Circular Right-shift with Carry (crc) places the Least Significant Bit (LSB) of R1 into C.
- Next, we check the value of C. If it is 0, goes back to address 11 to circulate again until C becomes a 1. When C = 1, control goes to address 13 to increment R2 and then returns to address 9 to check the content of R1 for an all 0's state.

Binary microprogram to count the number of 1's in R1

- The binary micro-program is given in Table 10-5.
- The 16 bits for the control word that selects the processor micro-operations are derived from Table 9-8.
- The multiplexer select bits select the inputs to the two multiplexers. Bit 17 is 0 in address 10 for selecting an external address. In all other cases, it is 1 to select the address field of the micro-instruction. When bits 18, 19, and 20 are 000, the next address is determined directly from the address field.
- When bits 18, 19, 20 are 011, they select the Z bit for MUX 2.
  - If Z = 1, an external address is transferred to CAR.
  - If Z = 0, CAR is incremented and the next address is the next one in sequence.

Control of Processor Unit

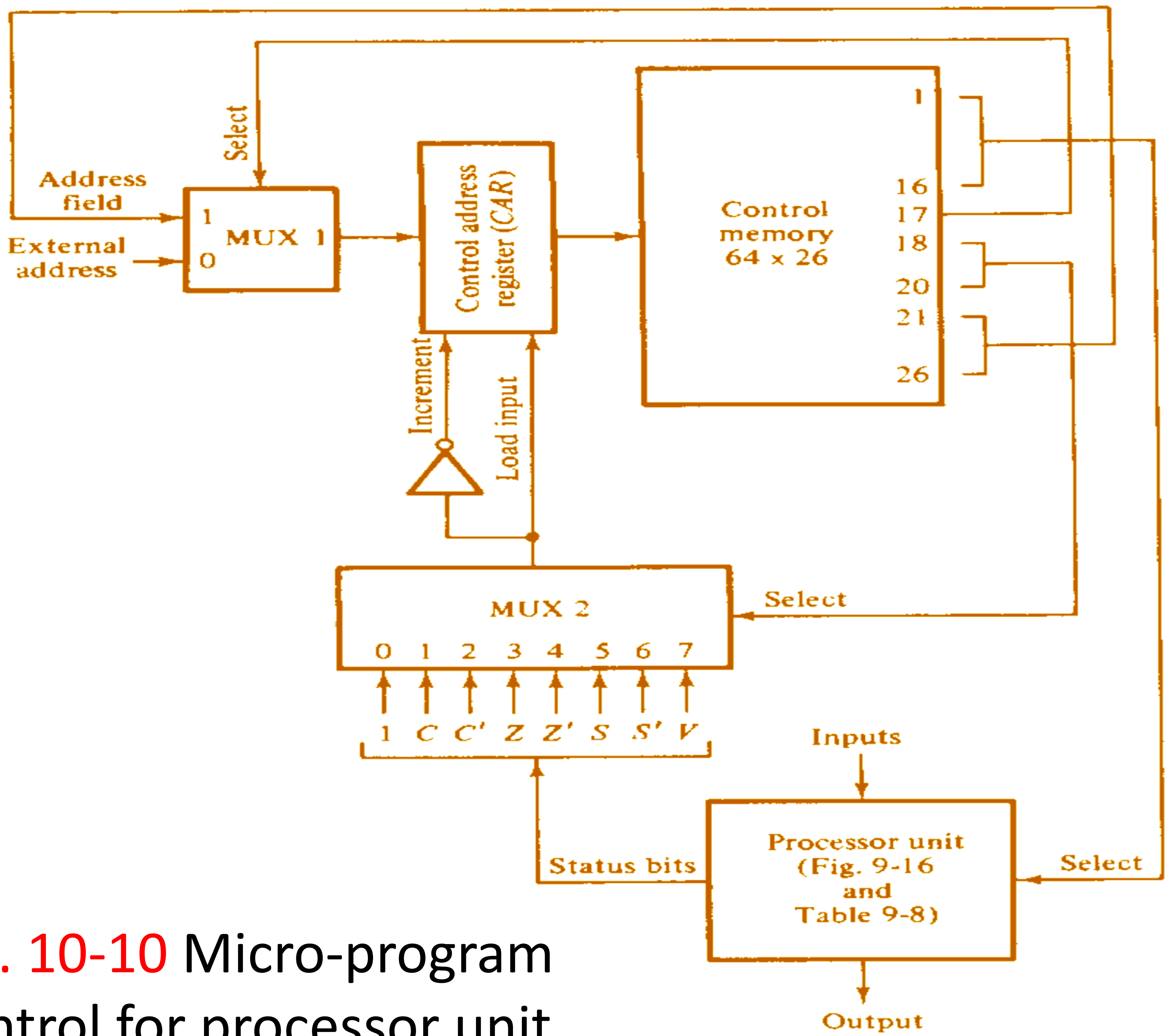
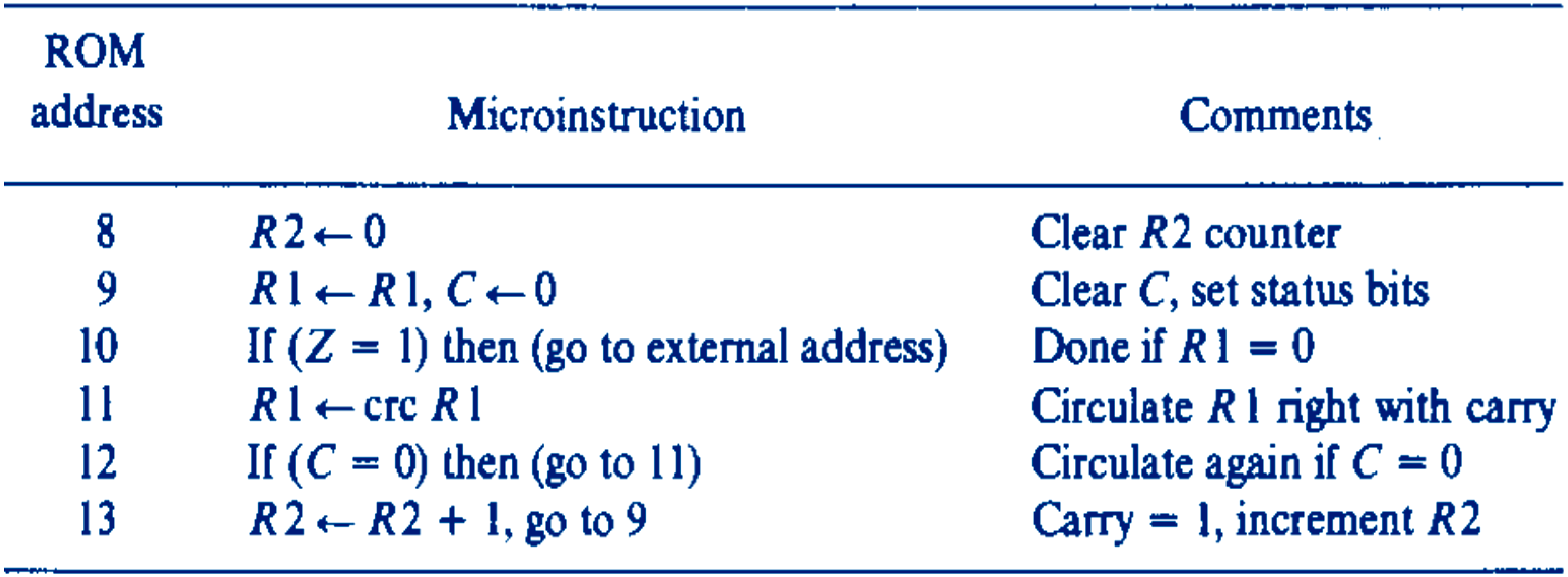


Fig. 10-10 Micro-program control for processor unit

Binary microprogram to count the number of 1’s in R1

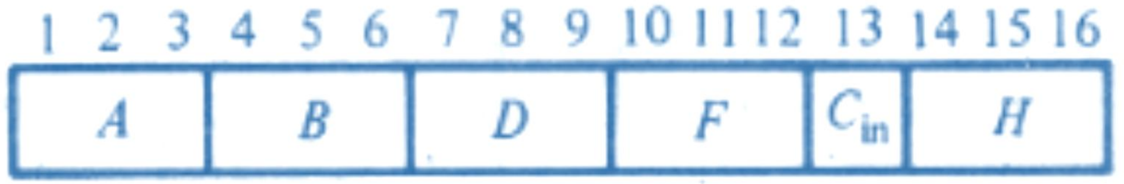
Table 10-4 Symbolic micro-program



ROM address	Microinstruction	Comments
8	$R2 \leftarrow 0$	Clear $R2$ counter
9	$R1 \leftarrow R1, C \leftarrow 0$	Clear C, set status bits
10	If $(Z = 1)$ then (go to external address)	Done if $R1 = 0$
11	$R1 \leftarrow$ crc $R1$	Circulate $R1$ right with carry
12	If $(C = 0)$ then (go to 11)	Circulate again if $C = 0$
13	$R2 \leftarrow R2 + 1$, go to 9	Carry = 1, increment $R2$

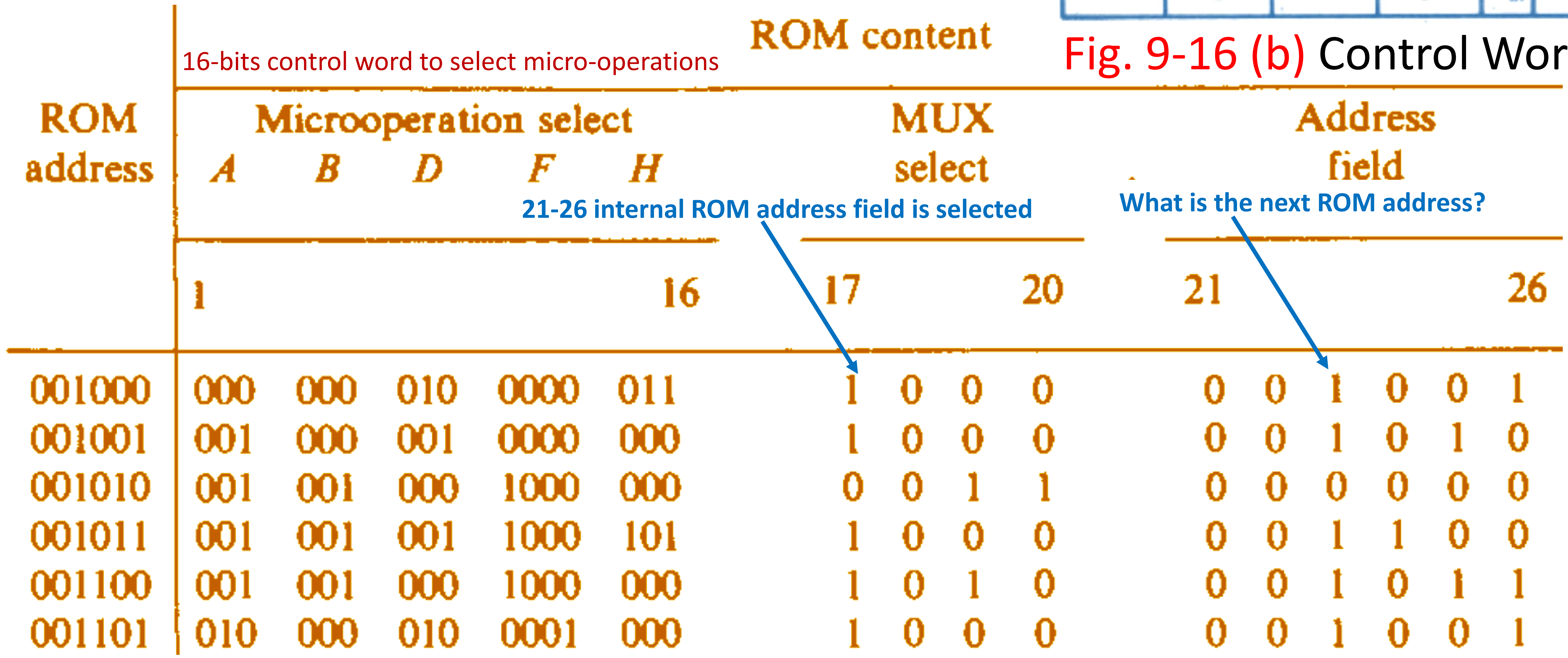
Binary microprogram to count the number of 1’s in R1

Table 10-5 Binary micro-program



1 2 3	4 5 6	7 8 9	10 11 12	13	14 15 16
A	*B*	*D*	*F*	C_{in}	*H*

Fig. 9-16 (b) Control Word



16-bits control word to select micro-operations

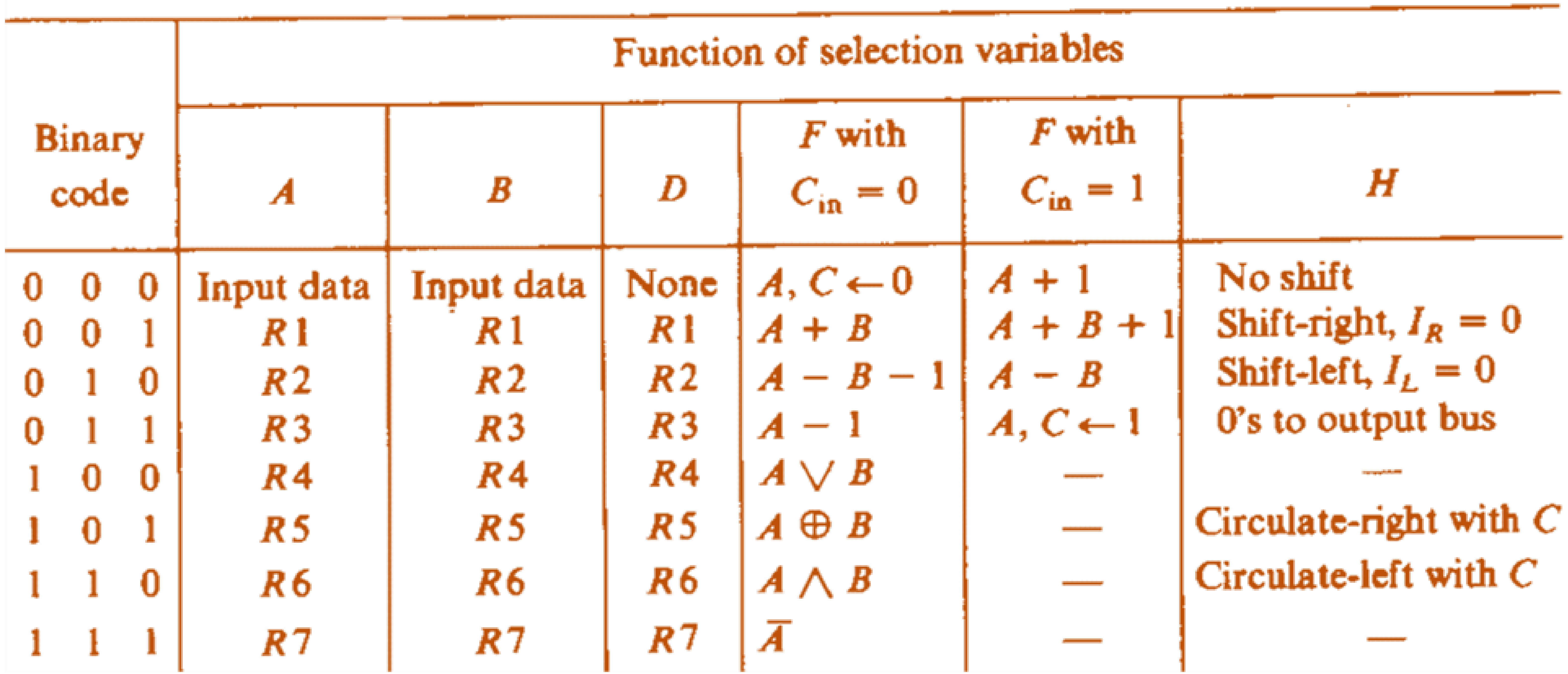
ROM content

21-26 internal ROM address field is selected

What is the next ROM address?

ROM address	Microoperation select *A* (1)	*B*	*D*	*F*	*H* (16)	MUX select (17–20)	Address field (21–26)
001000	000	000	010	0000	011	1 0 0 0	0 0 1 0 0 1
001001	001	000	001	0000	000	1 0 0 0	0 0 1 0 1 0
001010	001	001	000	1000	000	0 0 1 1	0 0 0 0 0 0
001011	001	001	001	1000	101	1 0 0 0	0 0 1 1 0 0
001100	001	001	000	1000	000	1 0 1 0	0 0 1 0 1 1
001101	010	000	010	0001	000	1 0 0 0	0 0 1 0 0 1

TABLE 9-8 Functions of control variables for the processor of Fig. 9-16

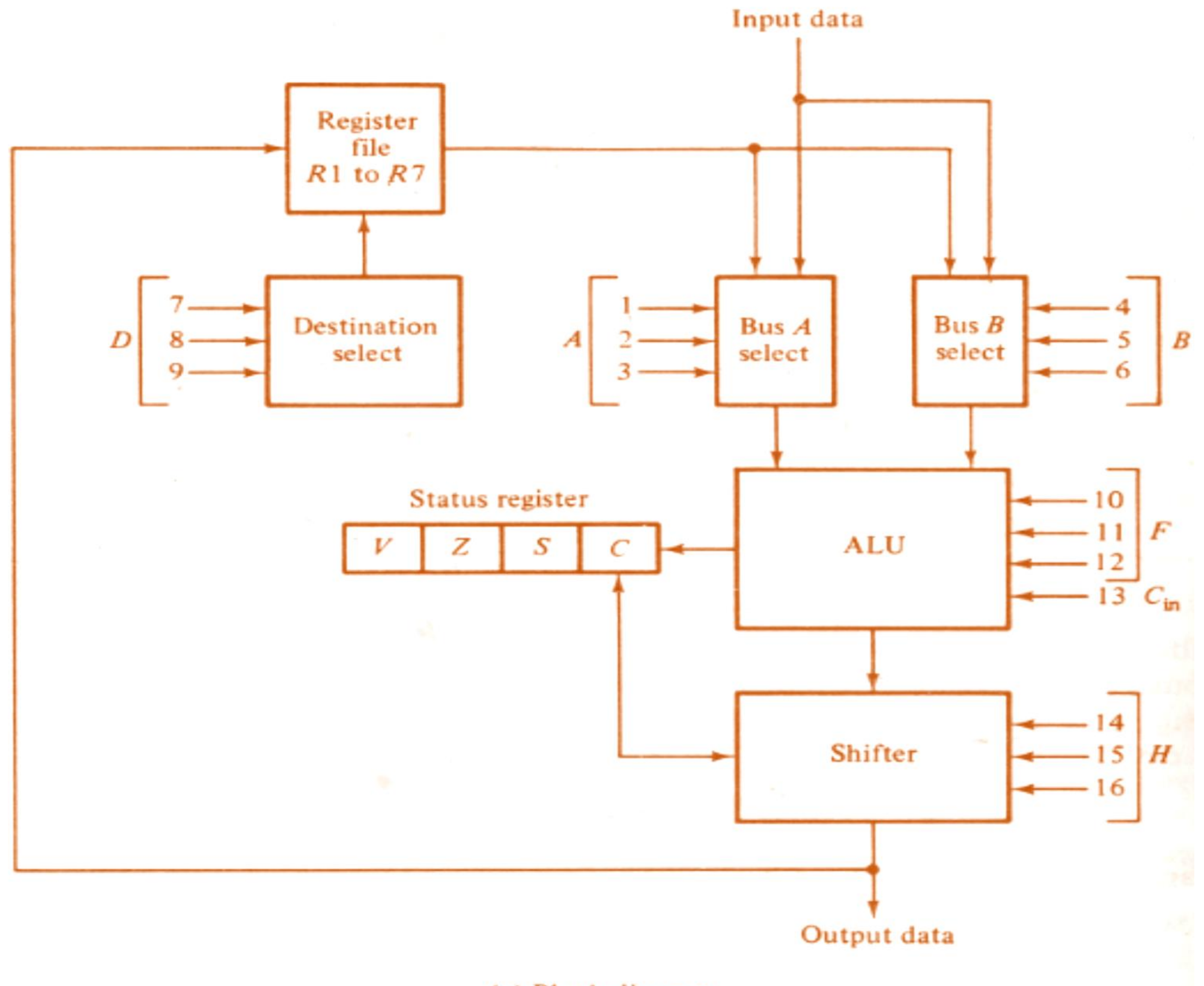


Binary code	Function of selection variables					
	A	B	D	F with $C_{in} = 0$	F with $C_{in} = 1$	H
0 0 0	Input data	Input data	None	$A, C \leftarrow 0$	$A + 1$	No shift
0 0 1	$R1$	$R1$	$R1$	$A + B$	$A + B + 1$	Shift-right, $I_R = 0$
0 1 0	$R2$	$R2$	$R2$	$A - B - 1$	$A - B$	Shift-left, $I_L = 0$
0 1 1	$R3$	$R3$	$R3$	$A - 1$	$A, C \leftarrow 1$	0's to output bus
1 0 0	$R4$	$R4$	$R4$	$A \vee B$	—	—
1 0 1	$R5$	$R5$	$R5$	$A \oplus B$	—	Circulate-right with C
1 1 0	$R6$	$R6$	$R6$	$A \wedge B$	—	Circulate-left with C
1 1 1	$R7$	$R7$	$R7$	$\bar{A}$	—	—

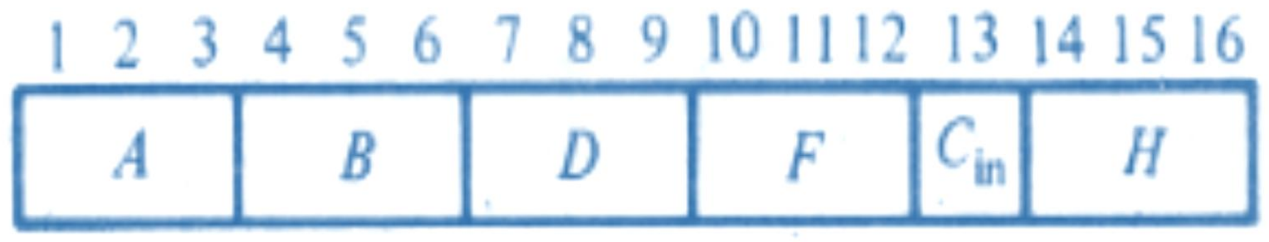
Binary microprogram to count the number of 1’s in R1

- The micro-instruction at address 12 selects the complement of the carry bit or C'.
  - If C = 0 then C'= 1, the address field (binary 1011) is transferred into CAR.
  - If C = 1 then C'= 0, the CAR is incremented to give 13 for the next address.
- The micro-program concept is a systematic procedure for designing the control unit of a digital system. Once the micro-instruction format is established, the design is done by writing a micro-program, which is similar to writing a program for a computer. For this reason, the micro-program method is sometimes referred to as **firmware** to distinguish it from the *hardware method* (which we called a *hard-wired control*) and the *software concept* which constitutes a *programming method*.

Processor Unit with Control Variable



(a) Block diagram



(b) Control word

Fig. 9-16

Thanks for attending….