AMERICAN INTERNAIONAL UNIVERSITY – BANGLADESH

Where leaders are created

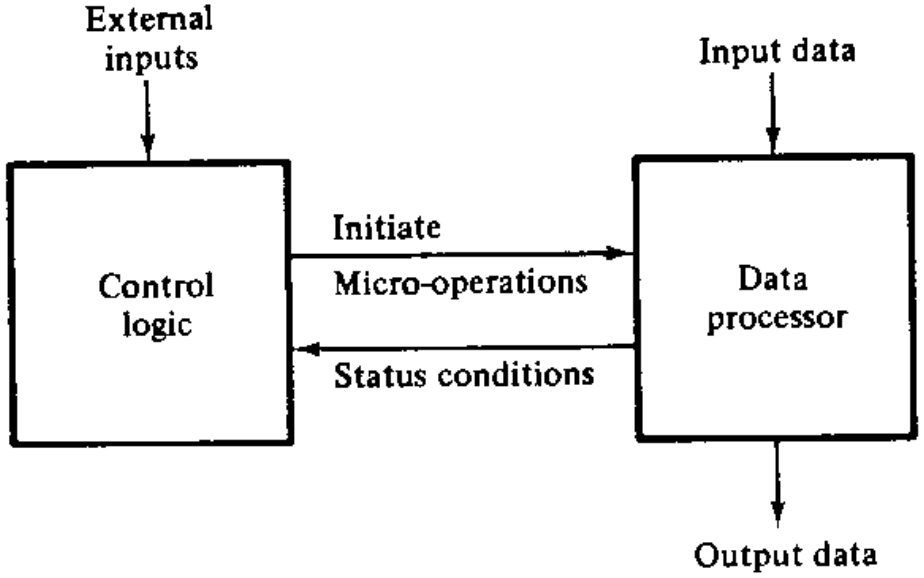
Control Logic Design

[Ch # 10]

Department of EEE

Introduction

- The process of logic design is a complex undertaking.
- The relationship between the control and the data processor in a digital system is shown in below figure.
- The data processor part may be general purpose processor unit or it may consist of individual registers.
- The control initiates all microoperations in the data processor.



Control Organization

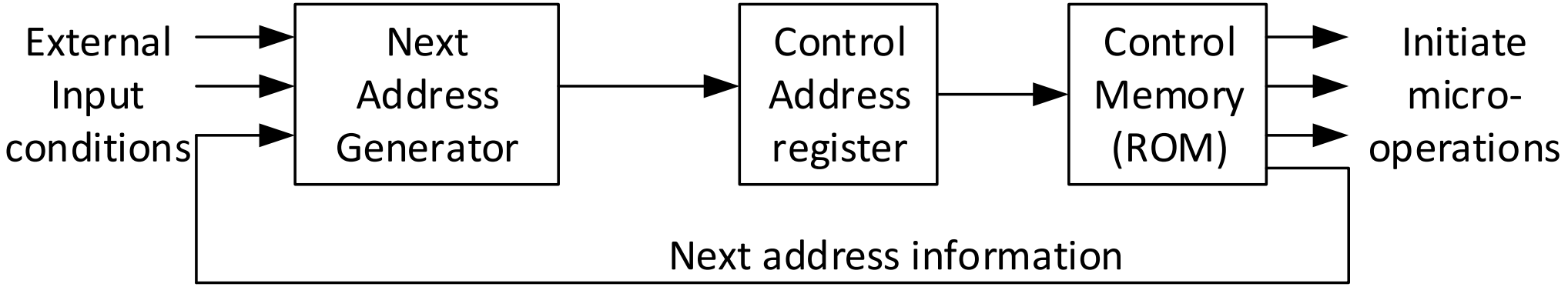
Once a control sequence has been established, the sequential system that implements the control operations must be designed.

- Methods of Control Organization (4 methods)
  - One flip-flop per state method
  - Sequence register and decoder method
  - PLA control
  - **Microprogram control**
- The first two method must use SSI and MSI circuits for the implementations.
- PLA or Microprogram uses an LSI device.

N.B: VLSI (Very large scale integration) Millions of transistors are integrated on a single silicon chip whereas in LSI (large scale integration) a few thousands of transistors are integrated on a single silicon chip

Microprogram Control

- The Purpose of the control unit is to initiate a series of sequential steps of microoperations.
- A control unit whose control variables are stored in a memory is called a **microprogrammed control unit**.
- Each control word of memory is called a **microinstruction.**
- A sequence of microinstructions is called a **microprogram.**



Hard-wired Control -Example

- The design is carried out in five consecutive steps

1. The problem is stated
2. An initial equipment configuration is assumed
3. An algorithm is formulated
4. The data-processor part is specified
5. The control logic is designed

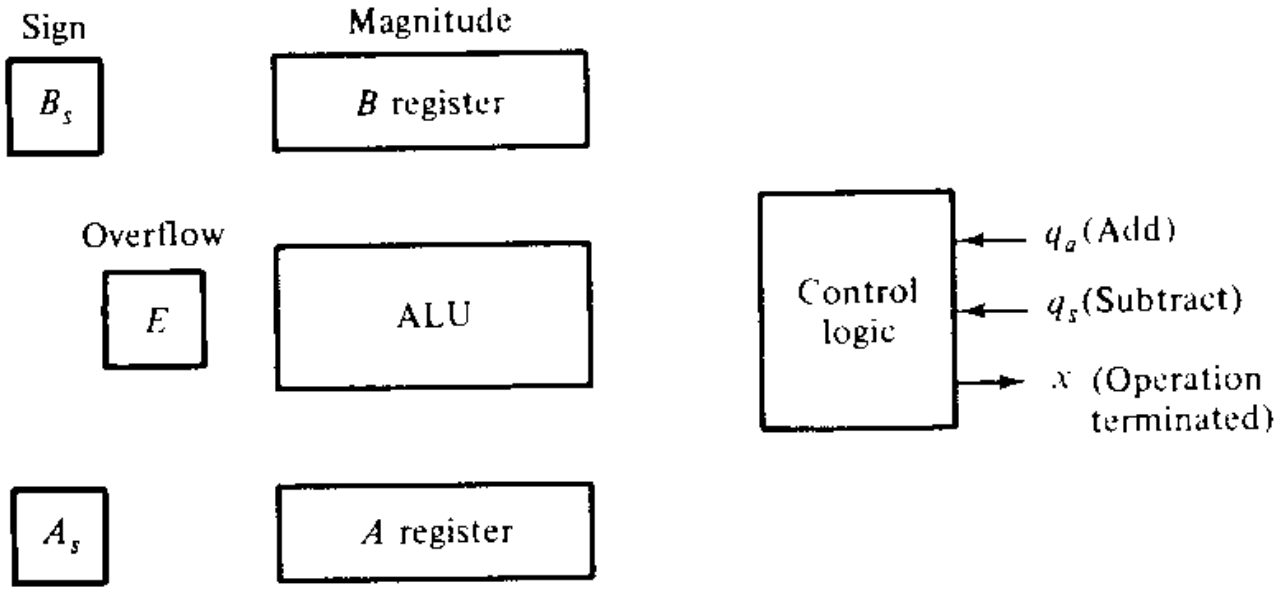
1. Statement of the problem

- Addition and subtraction of binary fixed point numbers when negative numbers are in sign 2's complement form.
- The addition of two numbers stored in registers of finite length may result in a sum that exceeds the storage capacity of the register by one bit.
- The extra bit is said to cause an overflow.

+ 6	0 000110		− 6	1 111010	
		+			+
+ 9	0 001001		+ 9	0 001001	
+ 15	0 001111		+ 3	0 000011	
+ 6	0 000110		− 9	1 110111	
		+			+
− 9	1 110111		− 9	1 110111	
− 3	1 111101		− 18	1 101110	

2.Equipment Configuration

- The two signed binary numbers to be added or subtracted contains n bits.
- The magnitudes of numbers contain k=n-1 bits are stored in registers A and B.
- The sign bits are stored in Flip Flops As and Bs.



Register configuration for the Adder-subtractor

3.Derivation of the Algorithm

- When the numbers are added or subtracted algebraically, there are eight different conditions to be consider and may be expressed as compact form as follows:

$$(\pm A) \pm (\pm B)$$

- In arithmetic operation specified in subtraction, we change the sign of B and add. So the relationships :

$$(\pm A) - (+B) = (\pm A) + (-B)$$
$$(\pm A) - (-B) = (\pm A) + (+B)$$

- This reduce the number of possible conditions to four, namely:

$$(\pm A) + (\pm B)$$

3. Derivation of the Algorithm cont….

- When the signs of A and B are the same, we add the two magnitudes and the sign of the result in the same as the common sign.
- When the sign of A and B are not same, we subtract the smaller number from the larger and the sign of the result is the sign of the larger number.

	if $A \geqslant B$	if $A < B$
$(+A) + (+B) =$	$+(A + B)$	
$(+A) + (-B) =$	$+(A - B)$	$-(B - A)$
$(-A) + (+B) =$	$-(A - B)$	$+(B - A)$
$(-A) + (-B) =$	$-(A + B)$	

Flowchart for sign magnitude addition and subtraction

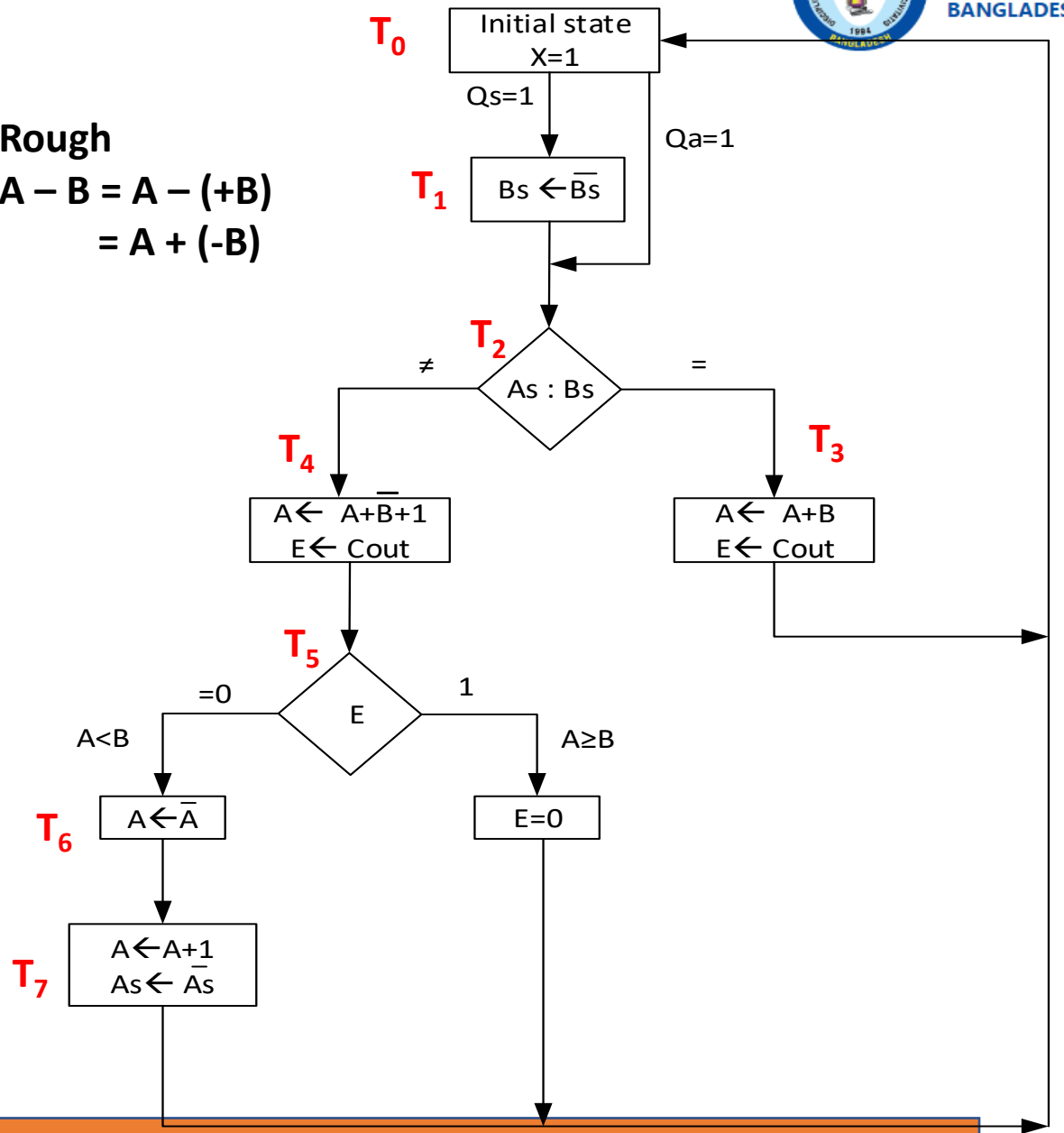
An operation is initiated by either input q_s or input q_a. Input q_s initiates a subtraction operation, so the sign of B is complemented. Input q_a initiates an add operation and the sign of B is unchanged. The next step is to compare the two signs. The decision block marked with As:Bs symbolizes this decision. If the signs are equal, we take the path marked by the symbol =. Otherwise , we take the path marked by ≠

For equal sign, the contents of A is added to the contents of B and the sum is transferred to A. The value of the end carry in this case is an overflow, so the E flip flop is made equal to output carry C_{out}. The circuit then goes to its initial state and output becomes 1. The sign of the result in this case is the same as the original sign of A_s so the sign bit is left unchanged.

Rough

A – B = A – (+B)

= A + (-B)



Flowchart for sign magnitude addition and subtraction contd..

The two magnitudes are **subtracted** if the signs are not the same. The subtraction of the magnitudes is done by adding A to the 2's complement of B. No overflow can occur if the numbers are subtracted, so E is cleared to 0. A '1' in E indicates that A>B and the number in A is the correct result. The sign of the result again is equal to the original value of As. A '0' in E indicates that A<B. For this case, it is necessary to form the 2's complement of the value in A and complement the sign in As. The 2's complement of A can be done with one microoperation, A← A'+1. However, we want to use the previously designed ALU which does not have the 2's complement operation. For this reason, the 2's complement is obtained from the complement and increment operations which are available in the ALU.

Question: Draw a flow chart of sign magnitude addition and subtraction of A & B, where Q_A= 1 for addition and Q_S =1 for subtraction, the result stored in A and carry out is stored in E after ALU operation, and A_s and B_s are the sign of A and B respectively.

Twist:

***P – Q or P + Q**

***Ps and Qs are the sign of P and Q**

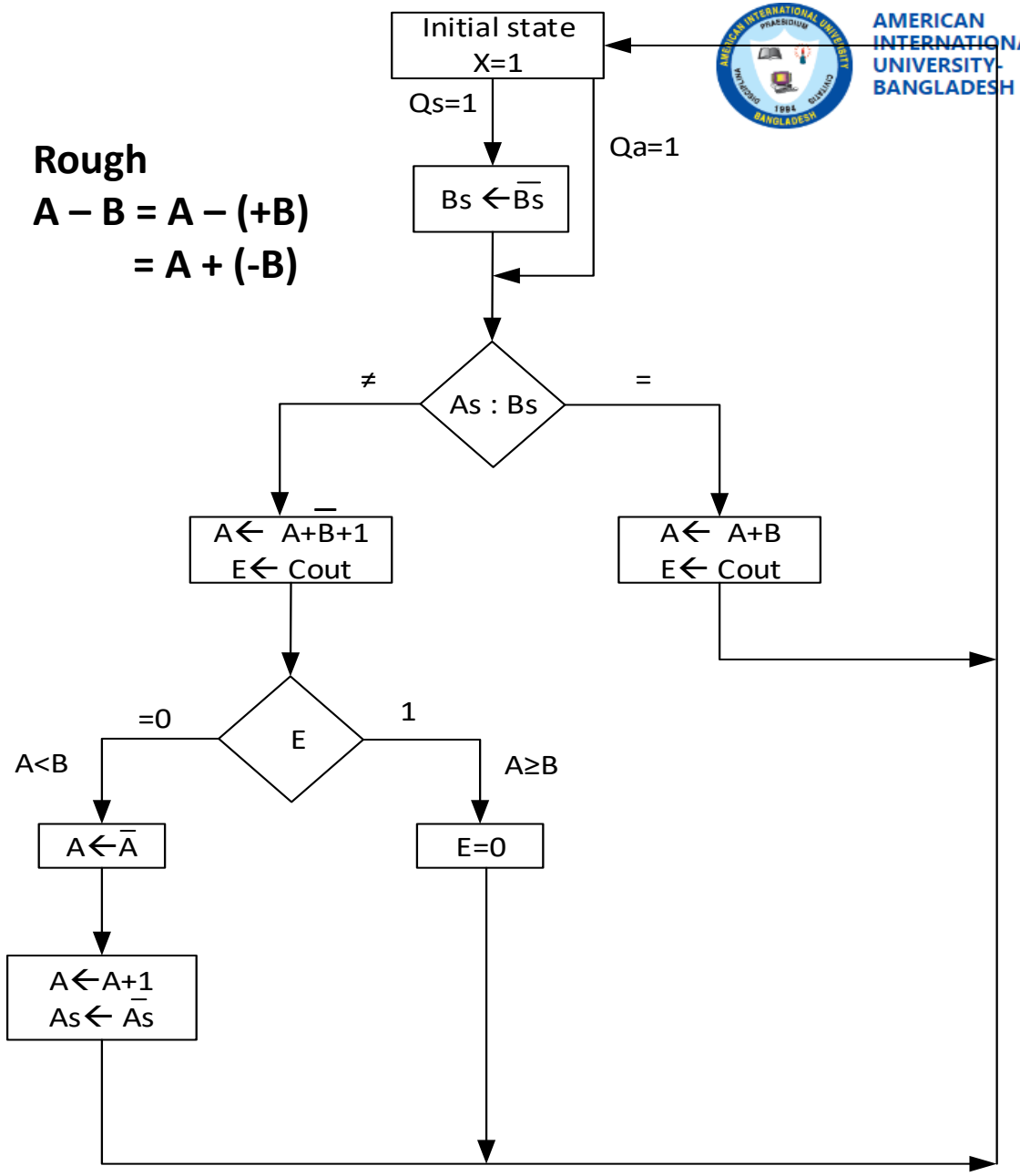
Example

Addition : +6 & +5
-6 & -5

Subtraction : +6 & +5
+5 & +6
-6 & -5
-5 & -6

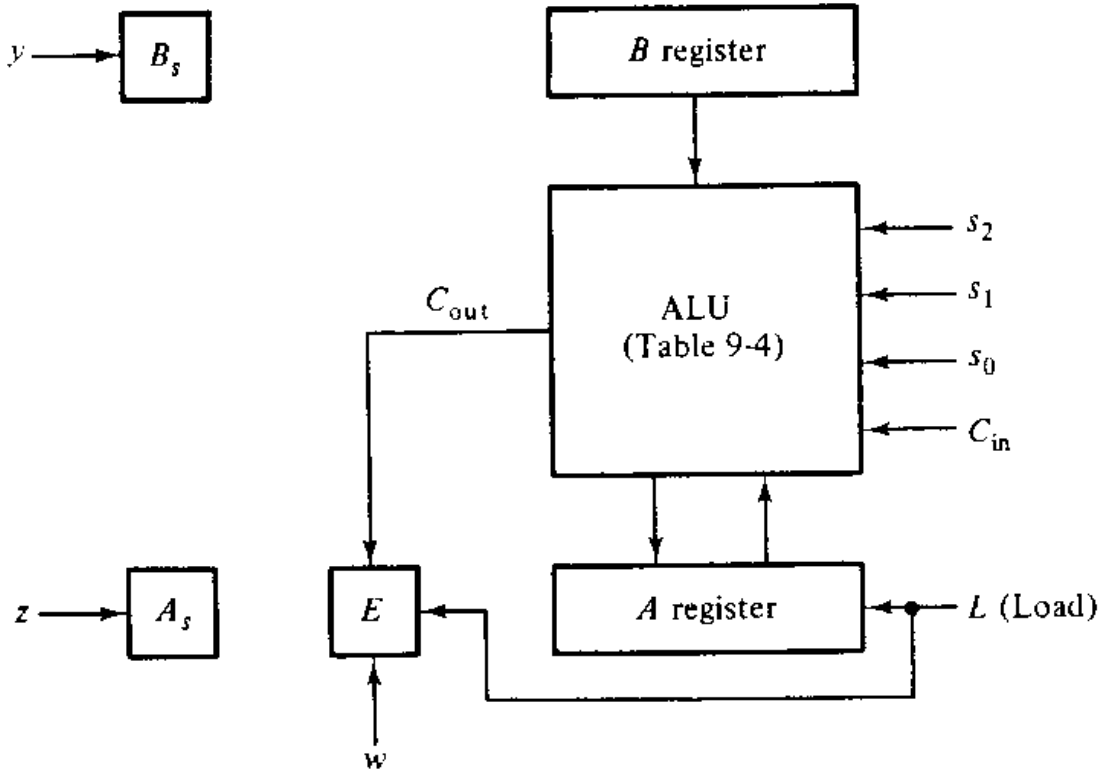
Rough

A – B = A – (+B)
= A + (-B)



4. Data Processor Register

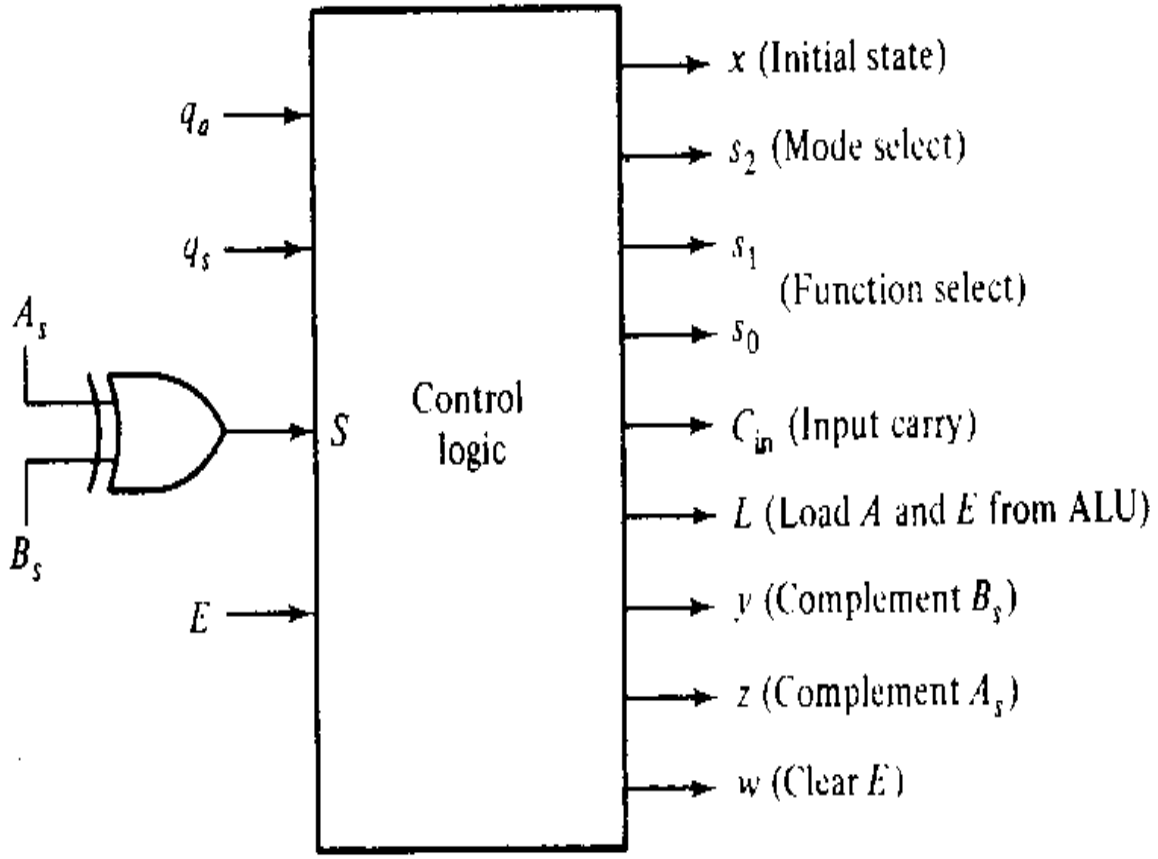
- The operations between A and B can be done with the ALU.
- The operations with As, Bs and E must be initiated with separate control variables.



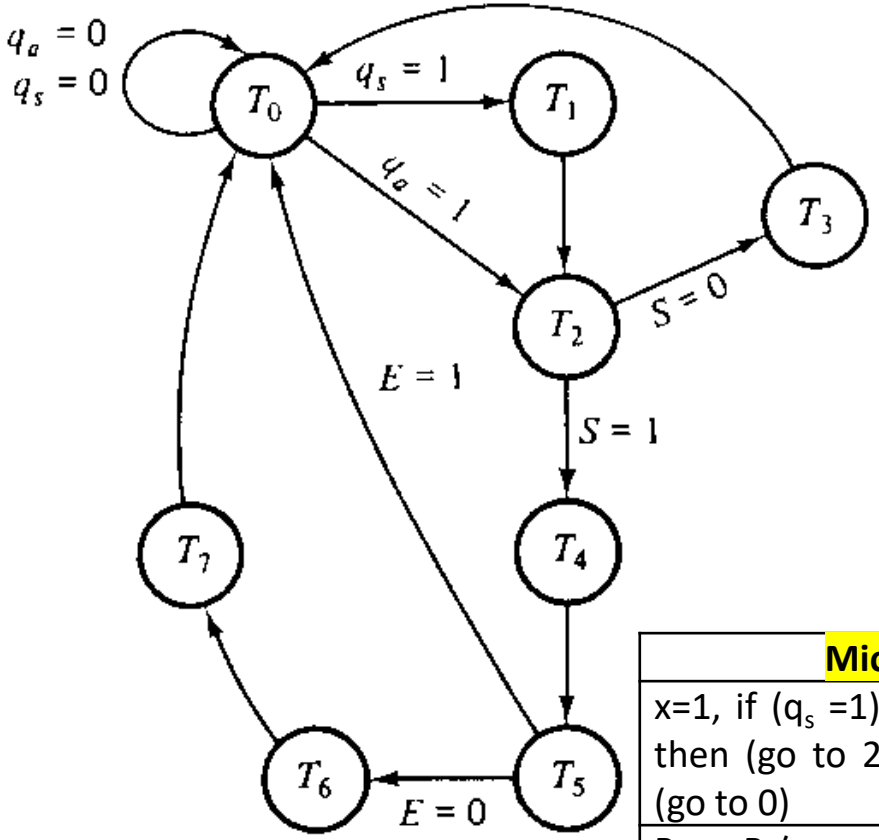
5. Control Block Diagram

- The control receives five inputs: two from the external environment and three from the data-processor.
- To simplify the design, we define new variable S:

S = As xor Bs



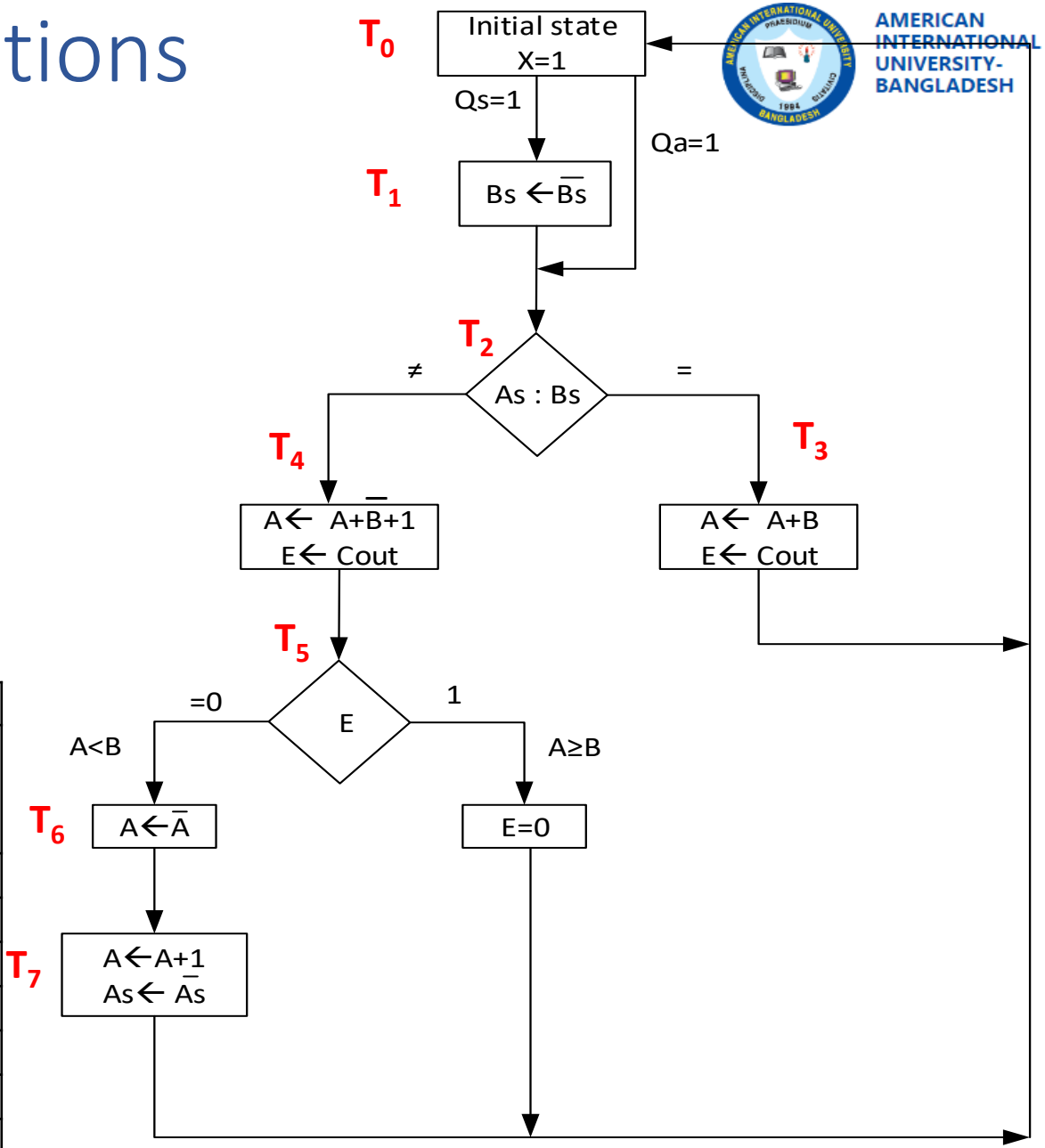
State Diagram and Microinstructions



State Diagram

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

Microinstructions
x=1, if (q_s =1) then (go to 1), If (q_a=1) then (go to 2), if (q_a and q_s = 0) then (go to 0)
Bs ← Bs'
If (S = 1), then go to 4
A← A+B, E←Cout, go to 0
A←A+B'+1, E←Cout
If(E = 1) then (go to 0), E←0
A←A'
A←A+1, As←As', go to 0



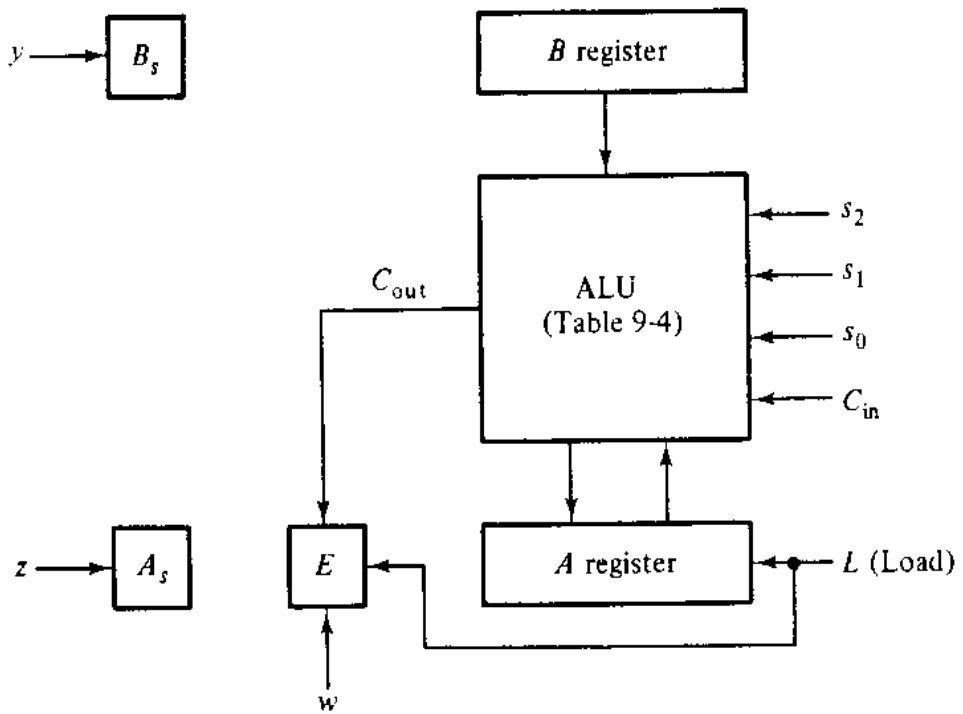
Sequence of register transfers

	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

Boolean Function

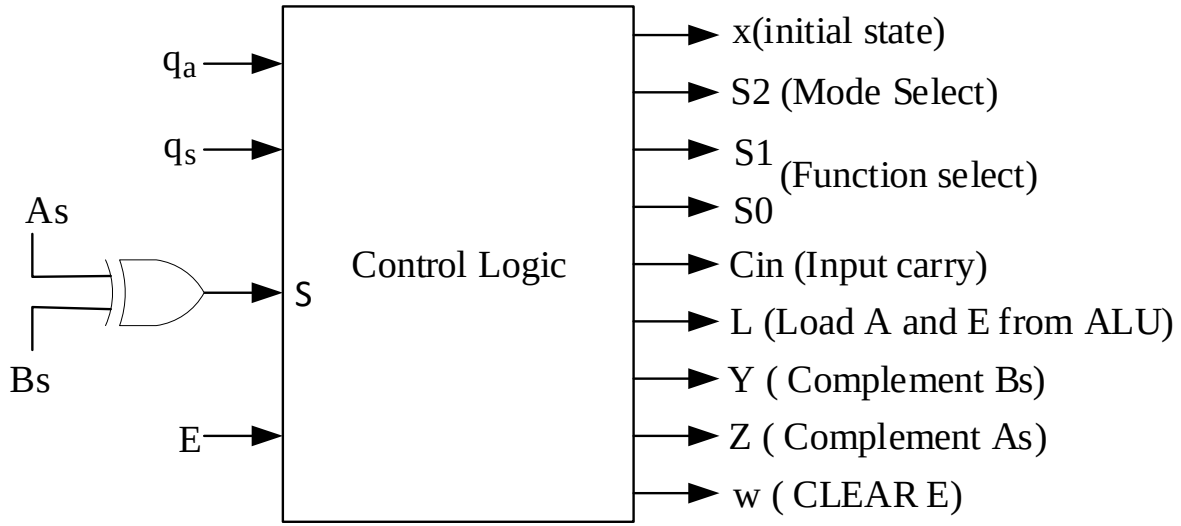
$x = T_0$
$s_2 = T_6$
$s_1 = T_4 + T_6$
$s_0 = T_3 + T_6$
$C_{in} = T_4 + T_7$
$L = T_3 + T_4 + T_6 + T_7$
$y = T_1$
$z = T_7$
$w = T_5$

Binary code	F with $C_{in} = 0$	F with $C_{in} = 1$
0 0 0	$A, C \leftarrow 0$	$A + 1$
0 0 1	$A + B$	$A + B + 1$
0 1 0	$A - B - 1$	$A - B$
0 1 1	$A - 1$	$A, C \leftarrow 1$
1 0 0	$A \vee B$	—
1 0 1	$A \oplus B$	—
1 1 0	$A \wedge B$	—
1 1 1	$\overline{A}$	—



To clear 'E', w pin needs to be activated

Q1. Design control state diagram for the previous flowchart. Develop control outputs for the signals given in following control logic block diagram using the Table and control states you defined.



Binary code	F with $C_{in} = 0$	F with $C_{in} = 1$
0 0 0	$A, C \leftarrow 0$	$A + 1$
0 0 1	$A + B$	$A + B + 1$
0 1 0	$A - B - 1$	$A - B$
0 1 1	$A - 1$	$A, C \leftarrow 1$
1 0 0	$A \vee B$	—
1 0 1	$A \oplus B$	—
1 1 0	$A \wedge B$	—
1 1 1	$\bar{A}$	—

Flow chart for counting the number of 1's in register R1 and storing in R2.

AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

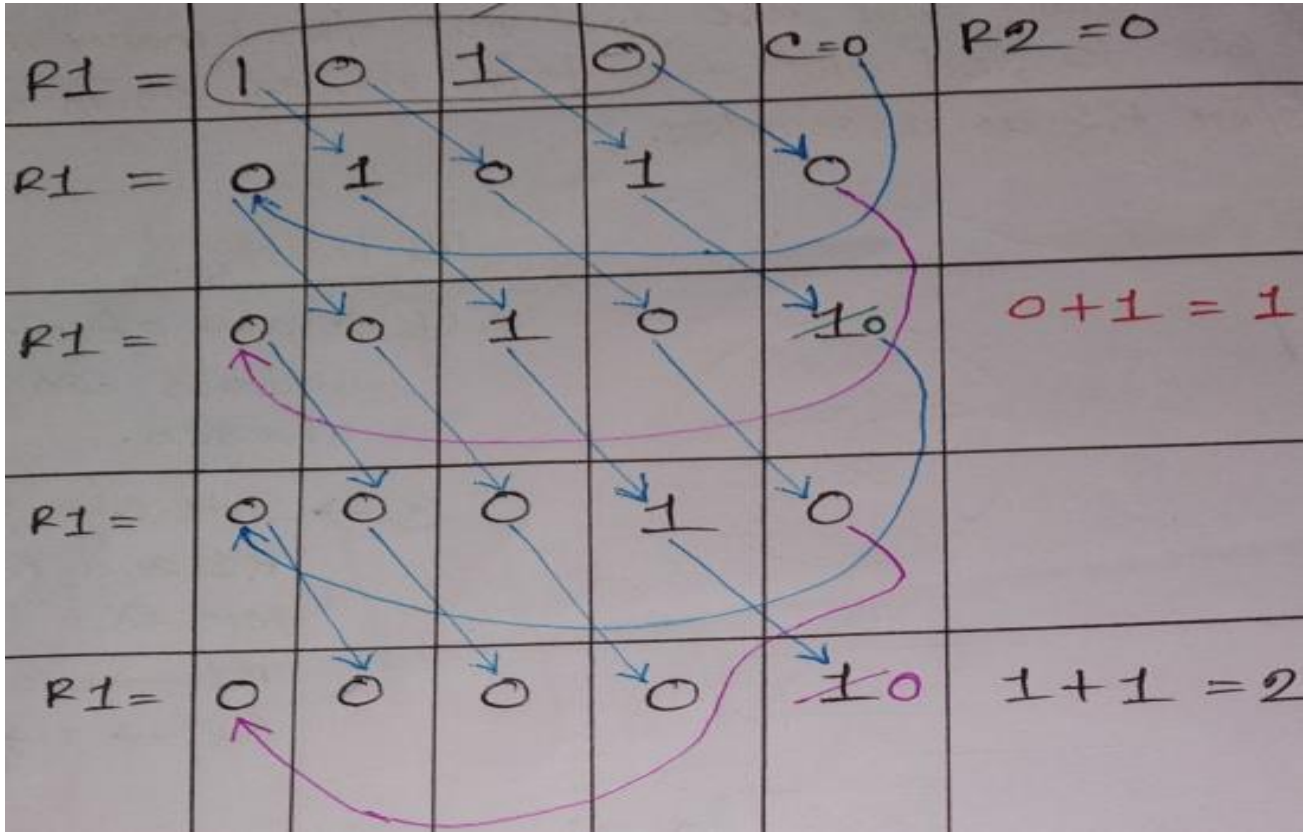
Example

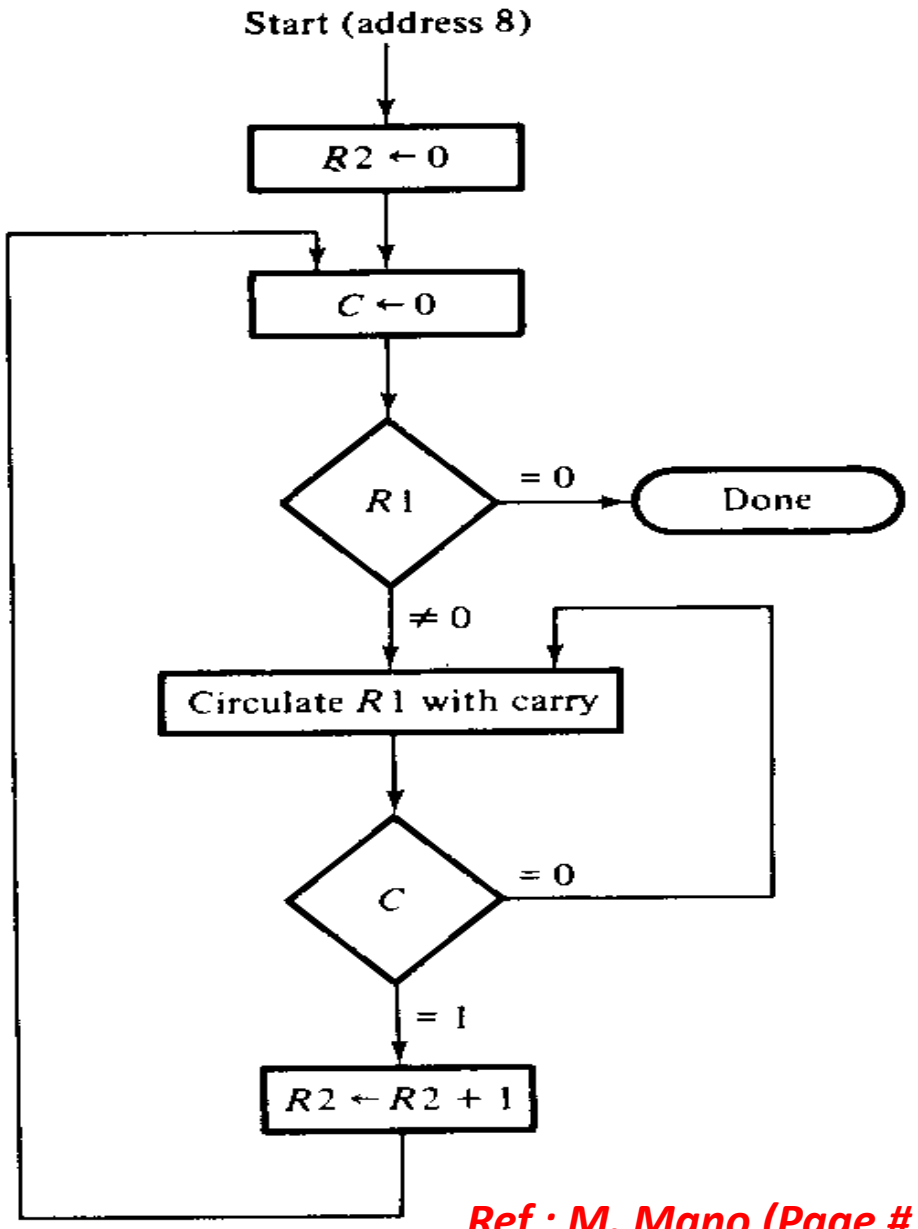
R1 : 1010 ------ two 1's

R2 : binary of two ---- 010

R1 → Register

R2 → Counter





Ref : M. Mano (Page # 432)

Homework

AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Q1. Draw the flow chart to find out the total number of 1's in register, R1. Use register R2 as counter. [See previous example]

Q2. Draw the flow chart to find out the total number of 1's in register, R2. Use register R1 as counter.

Q3. Draw the flow chart to find out the total number of 0's in register, R1. Use register R2 as counter.

Twist:
***Find total numbers of zeros**
***Register, counter interchanged**