Practice Set 1

COA

Solved by:

आचार्य बृजेन्द्र कुमार (BRAJ SIR)

"चार चवन्नी घोड़े पे, COA मेरे _____ "

Join for More:
https://chat.whatsapp.com/CjhLnhqPByLIIQIDi12GeL

YouTube: https://youtube.com/@roomno547

Practice Set 1

COA

Objective type question:

1. CLE (Clear E) & CME (Complement E)
2. Computer Organization
3. High impedance state
4. Memory Reference Instructions, Register Reference Instructions, I/O Instructions
5. 000(0) to 110(6).
6. (c) I/O
7. (c) 1000
8. Hold the address of instruction
9. (b) IR
10. Register Transfer language

Short Answer:

Answer 1:

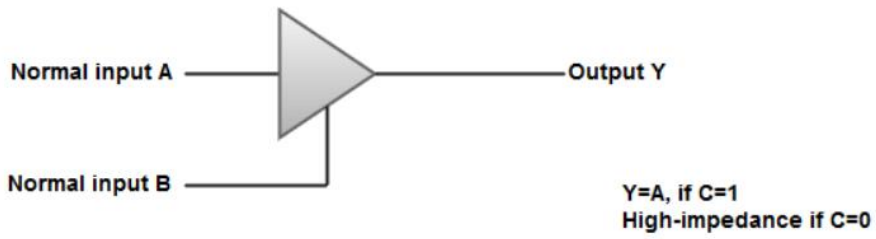
The symbolic notation used to describe the micro-operation transfers amongst registers is called Register Transfer language

The term Register Transfer means the availability of hardware logic circuit that can perform a stated micro-operation and transfer the result of the operation to the same or another register.

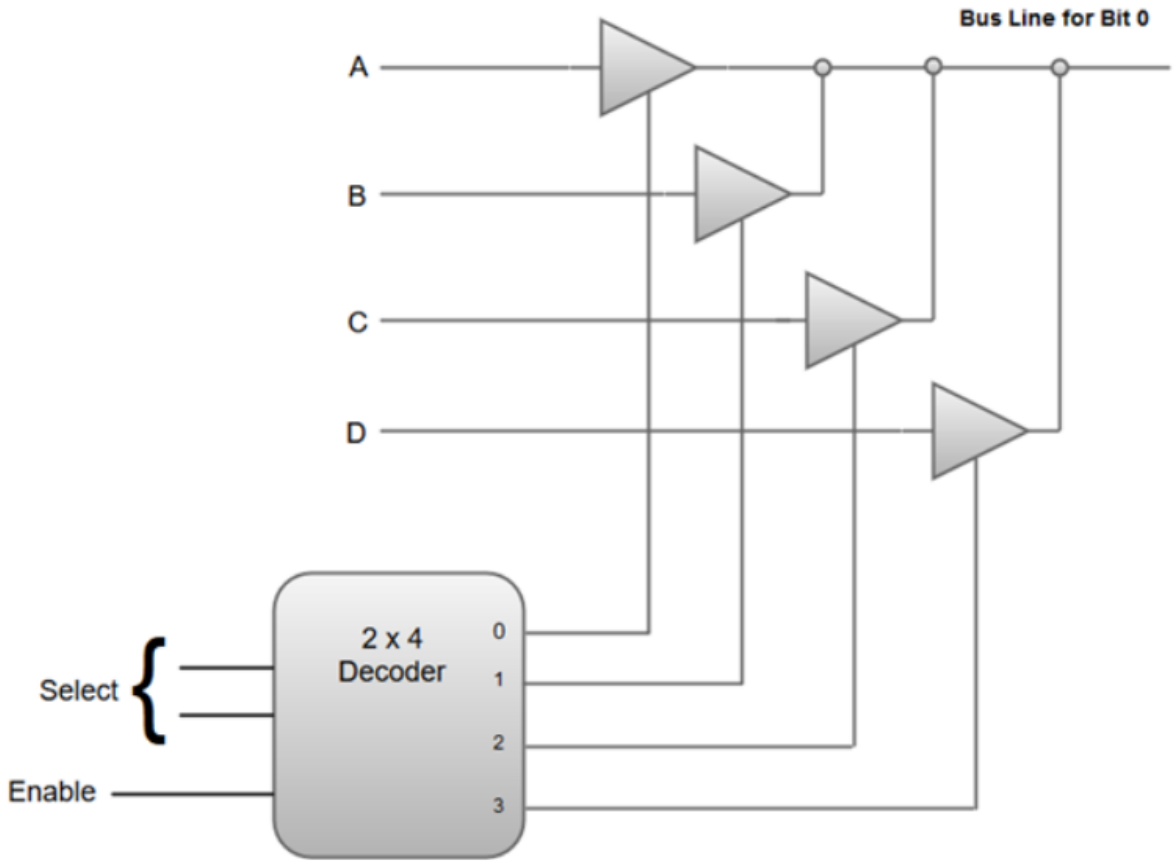
Example: R1 <- R2

P: R2 <- R1

Answer 2:



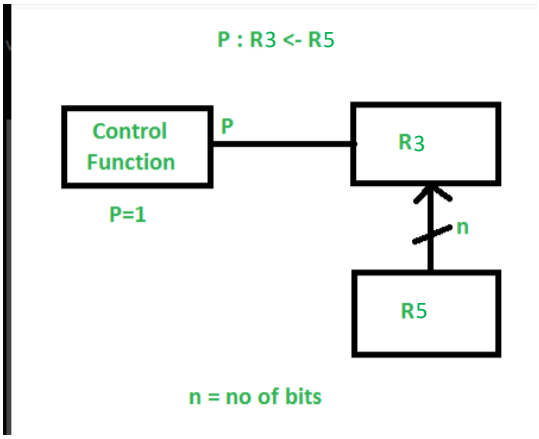
Bus line with three state buffer:



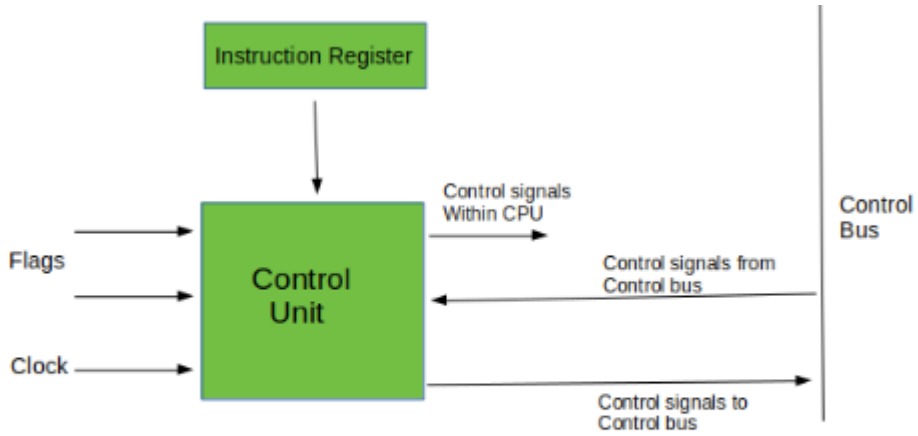
- The outputs generated by the four buffers are connected to form a single bus line.
- Only one buffer can be in active state at a given point of time.
- The control inputs to the buffers determine which of the four normal inputs will communicate with the bus line.
- A 2 * 4 decoder ensures that no more than one control input is active at any given point of time.

Answer 3:

It indicates that if P=1, then the content of R5 is transferred to R3. It is a unidirectional operation.



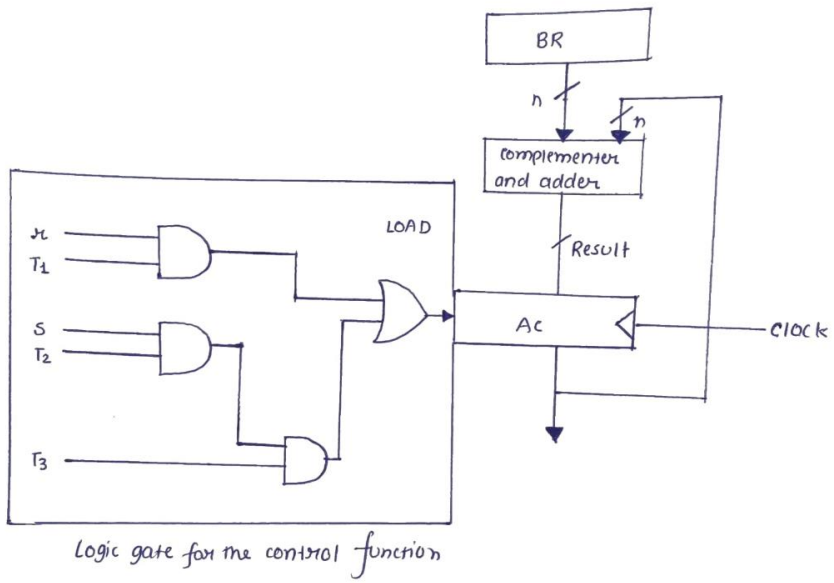
Answer 4:



Block Diagram of the Control Unit

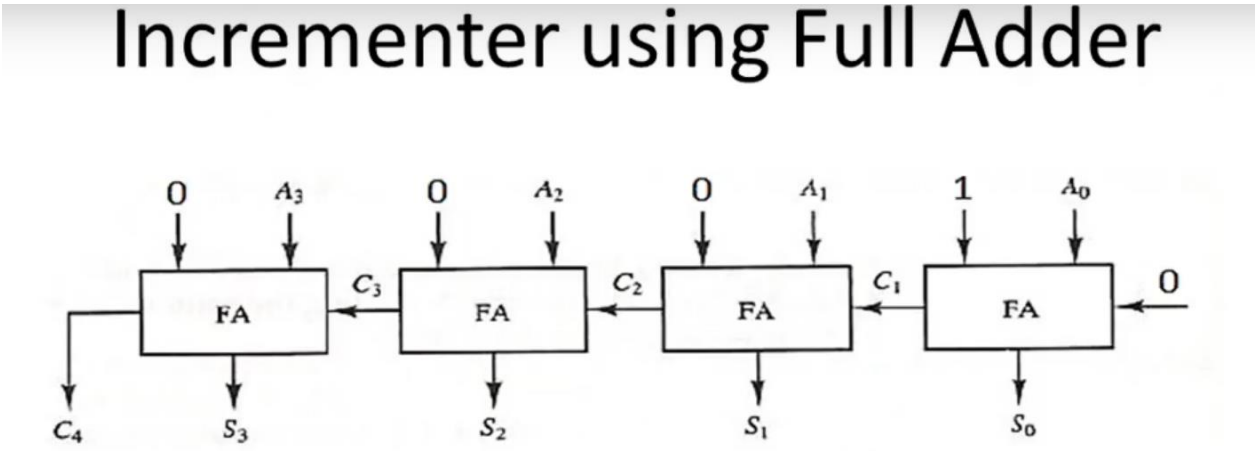
- It coordinates the sequence of data movements into, out of, and between a processor's many sub-units.
- It interprets instructions.
- It controls data flow inside the processor.
- It receives external instructions or commands to which it converts to sequence of control signals.
- It controls many execution units(i.e. ALU, data buffers and registers) contained within a CPU.
- It also handles multiple tasks, such as fetching, decoding, execution handling and storing results.

Answer 5:



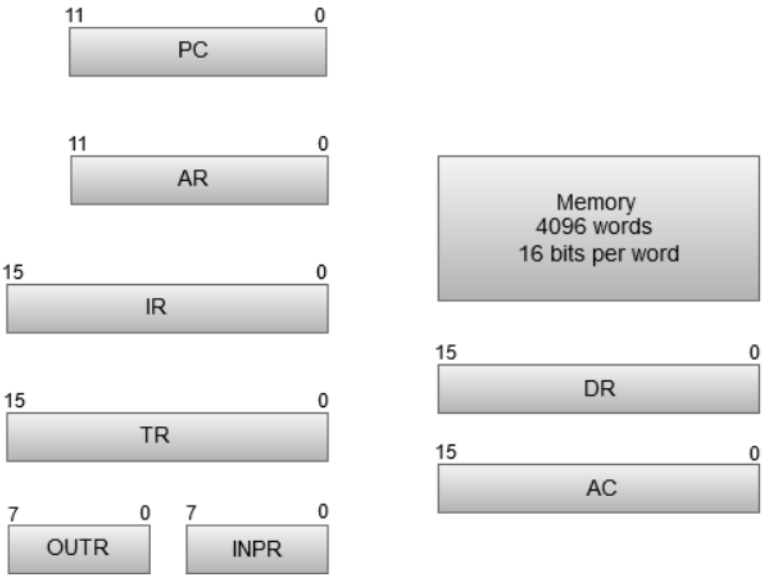
Logic gate for the control function

Answer 6:



Answer 7:

Register and Memory Configuration of a basic computer:



Register	Symbol	Number of bits	Function
Data register	DR	16	Holds memory operand
Address register	AR	12	Holds address for the memory
Accumulator	AC	16	Processor register
Instruction register	IR	16	Holds instruction code
Program counter	PC	12	Holds address of the instruction
Temporary register	TR	16	Holds temporary data
Input register	INPR	8	Carries input character
Output register	OUTR	8	Carries output character

Answer 8:

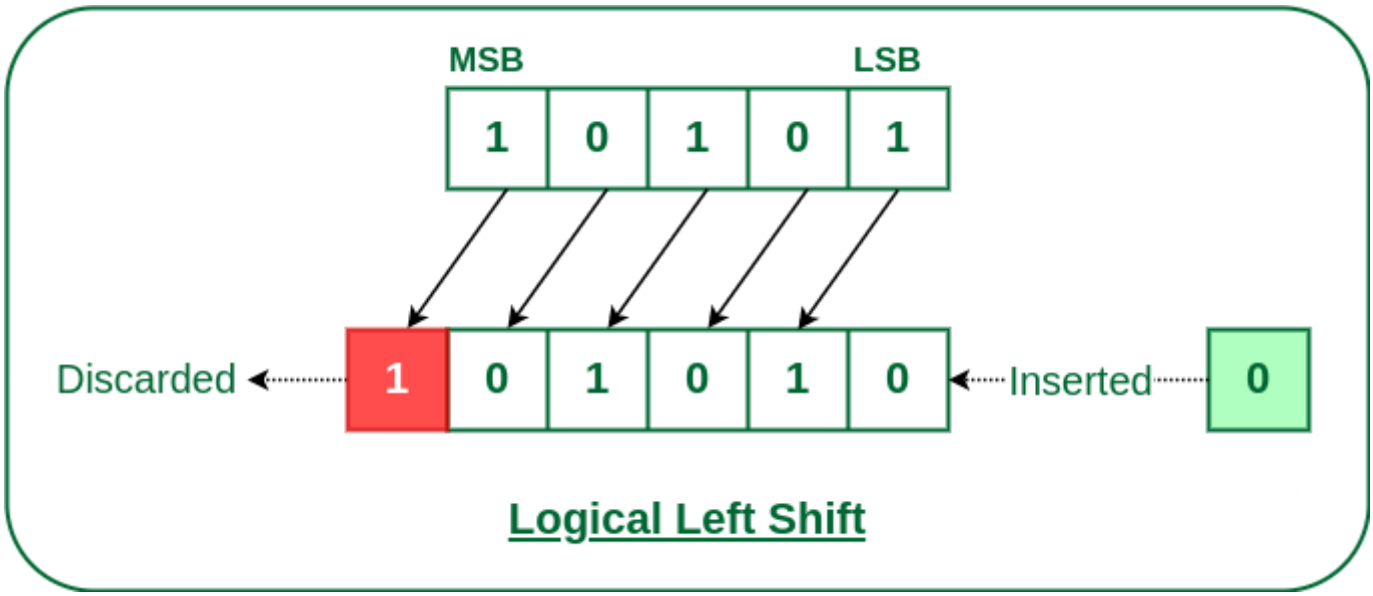
1. Logical Shift:

It transfers the 0 zero through the serial input. We use the symbols '**<<**' for the logical left shift and '**>>**' for the logical right shift.

Logical Left Shift:

In this shift, one position moves each bit to the left one by one. The Empty least significant bit (LSB) is filled with zero (i.e, the serial input), and the most significant bit (MSB) is rejected.

The left shift operator is denoted by the double left arrow key (<<). The general syntax for the left shift is shift-expression << k.



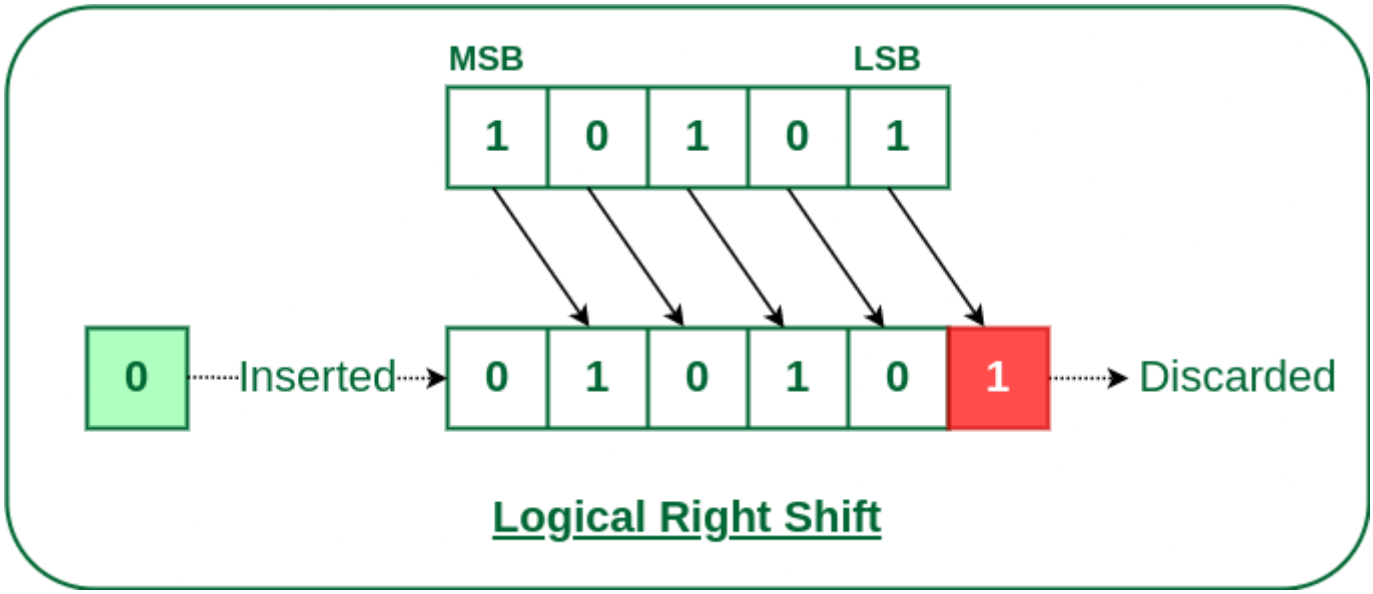
Logical Left Shift

Note: Every time we shift a number towards the left by 1 bit it multiplies that number by 2.

Logical Right Shift

In this shift, each bit moves to the right one by one and the least significant bit(LSB) is rejected and the empty MSB is filled with zero.

The right shift operator is denoted by the double right arrow key (>>). The general syntax for the right shift is "shift-expression >> k".



Logical Right Shift

Note: Every time we shift a number towards the right by 1 bit it divides that number by 2.

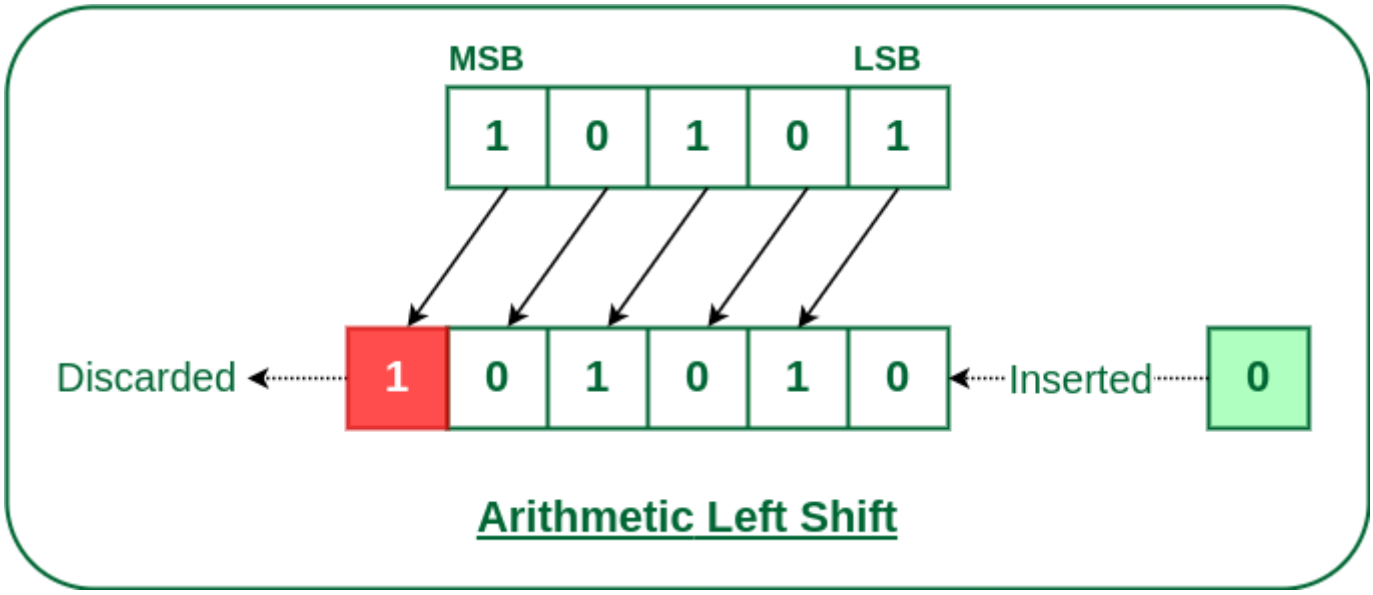
2. Arithmetic Shift:

The arithmetic shift micro-operation moves the signed binary number either to the left or to the right position.
Following are the two ways to perform the arithmetic shift.

1. Arithmetic Left Shift
2. Arithmetic Right Shift

Arithmetic Left Shift:

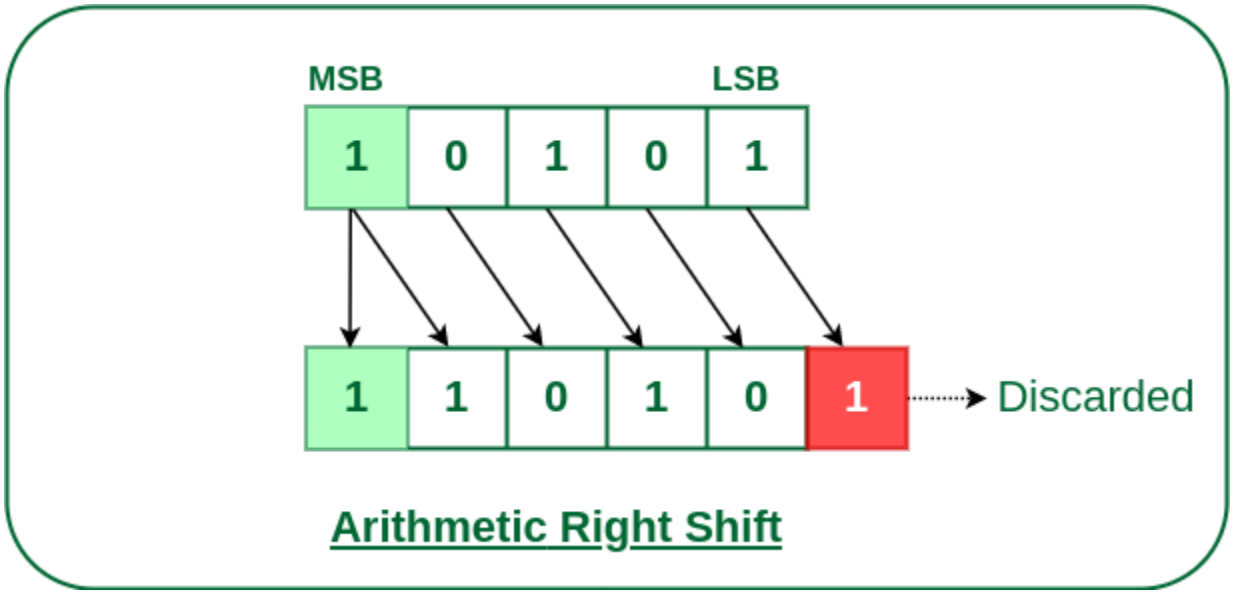
In this shift, each bit is moved to the left one by one. The empty least significant bit (LSB) is filled with zero and the most significant bit (MSB) is rejected. Same as the Left Logical Shift.



Arithmetic Left Shift

Arithmetic Right Shift:

In this shift, each bit is moved to the right one by one and the least significant(LSB) bit is rejected and the empty most significant bit(MSB) is filled with the value of the previous MSB.



Arithmetic Right Shift

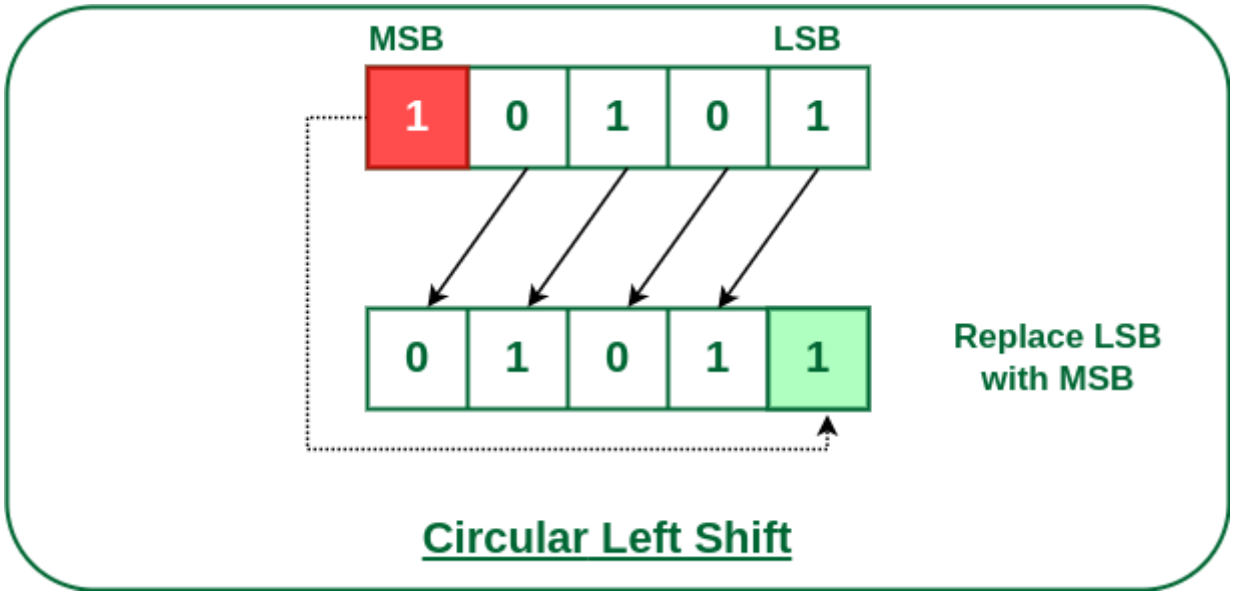
3. Circular Shift:

The circular shift circulates the bits in the sequence of the register around both ends without any loss of information.
Following are the two ways to perform the circular shift.

1. Circular Shift Left
2. Circular Shift Right

Circular Left Shift:

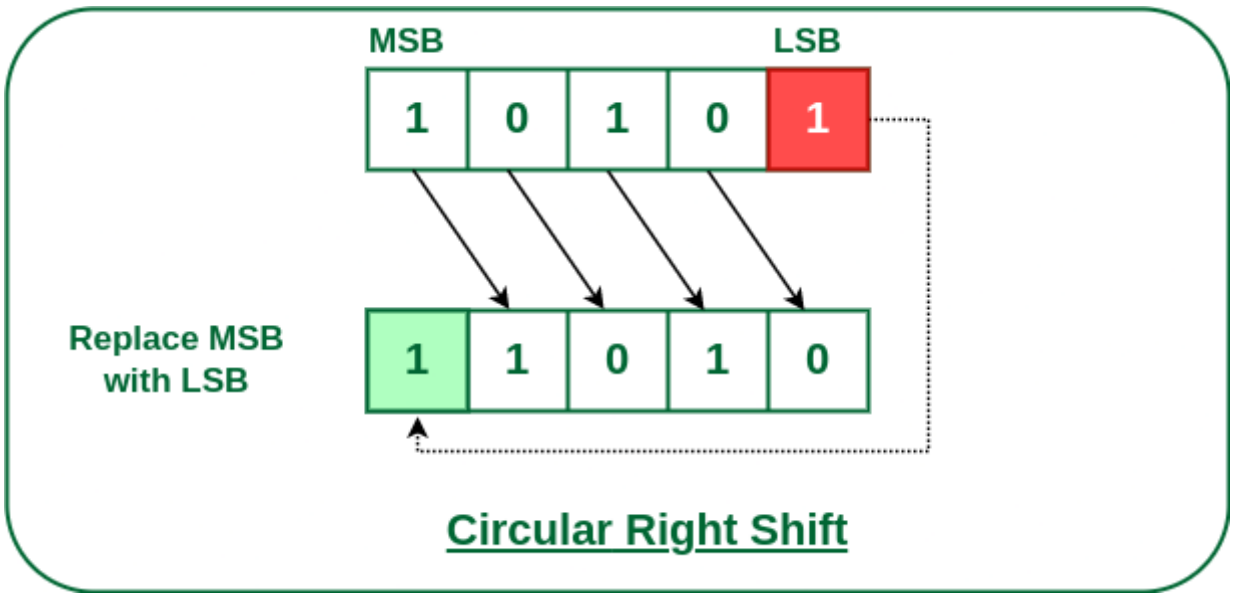
In this micro shift operation each bit in the register is shifted to the left one by one. After shifting, the LSB becomes empty, so the value of the MSB is filled in there.



Circular Left Shift

Circular Right Shift:

In this micro shift operation each bit in the register is shifted to the right one by one. After shifting, the MSB becomes empty, so the value of the LSB is filled in there.

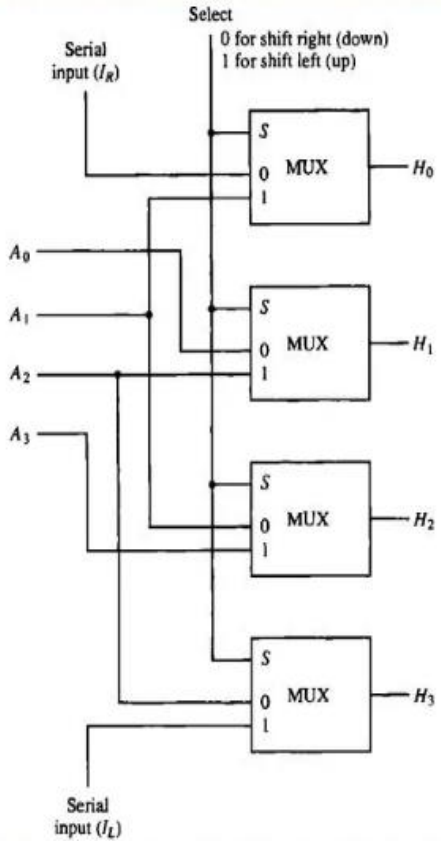


Circular Right Shift

List of shift microoperations

Symbolic designation	Description
R ← shl R	Shift-left register R
R ← shr R	shift-right register R
R ← cil R	circular shift-left register R
R ← cir R	circular shift-right register R
R ← ashl R	Arithmetic shift-left R
R ← ashr R	Arithmetic shift-right R

4-bits combinational circuit shifter



Function table

Select	Output			
S	H_0	H_1	H_2	H_3
0	I_R	A_0	A_1	A_2
1	A_1	A_2	A_3	I_L

Answer 9:

	1	0	0	0	1	1	0	1
Logical Shift left	0	0	0	1	1	0	1	0
Circular shift right	0	0	0	0	1	1	0	1
Logical shift right	0	0	1	0	1	1	1	0
Circular shift left	0	1	0	1	1	1	0	0

Answer 10:

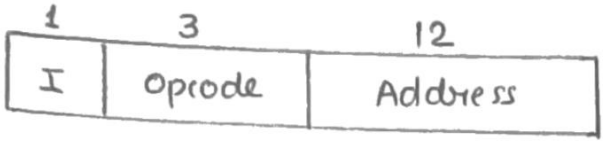
Solution - 10

$4K = 4 \times 1024 = 2^2 \times 2^{10} = 2^{12}$

Address : 12 bit

Indirect bit : 1 bit

Opcode bit = $16 - (12+1) = 16-13 = 3$ bit



Answer 11:

- The hardware implementation of logic microoperations requires that logic gates be inserted for each bit or pair of bits in the registers to perform the required logic function.
- Although there are 16 logic microoperations, most computers use only four-AND, OR, XOR (exclusive-OR), and complement from which all others can be derived.
- Figure below shows one stage of a circuit that generates the four basic logic microoperations.

- It consists of four gates and a multiplexer. Each of the four logic operations is generated through a gate that performs the required logic.
- The outputs of the gates are applied to the data inputs of the multiplexer. The two selection inputs S1 and S0 choose one of the data inputs of the multiplexer and direct its value to the output.
- The diagram shows one typical stage with subscript i. For a logic circuit with n bits, the diagram must be repeated n times for i = 0, 1, 2, ... , n - 1.
- The selection variables are applied to all stages. The function table in Fig. below lists the logic rnicrooperations obtained for each combination of the selection variables.

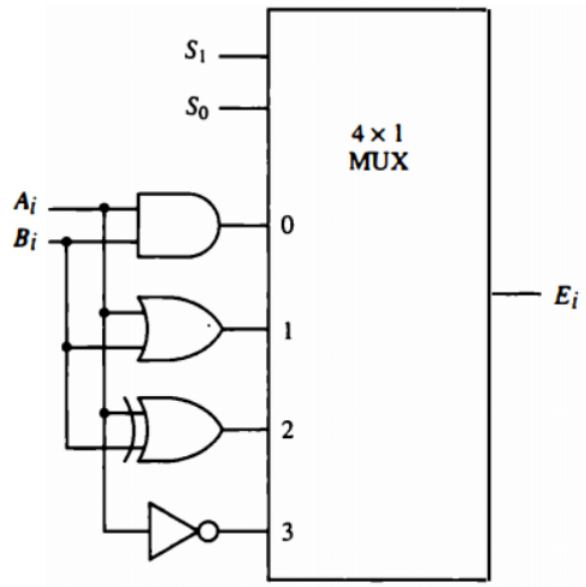
TABLE Truth Tables for 16 Functions of Two Variables

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

TABLE Sixteen Logic Microoperations

Boolean function	Microoperation	Name
$F_0 = 0$	$F \leftarrow 0$	Clear
$F_1 = xy$	$F \leftarrow A \wedge B$	AND
$F_2 = xy'$	$F \leftarrow A \wedge \overline{B}$	
$F_3 = x$	$F \leftarrow A$	Transfer A
$F_4 = x'y$	$F \leftarrow \overline{A} \wedge B$	
$F_5 = y$	$F \leftarrow B$	Transfer B
$F_6 = x \oplus y$	$F \leftarrow A \oplus B$	Exclusive-OR
$F_7 = x + y$	$F \leftarrow A \vee B$	OR
$F_8 = (x + y)'$	$F \leftarrow \overline{A \vee B}$	NOR
$F_9 = (x \oplus y)'$	$F \leftarrow \overline{A \oplus B}$	Exclusive-NOR
$F_{10} = y'$	$F \leftarrow \overline{B}$	Complement B
$F_{11} = x + y'$	$F \leftarrow A \vee \overline{B}$	
$F_{12} = x'$	$F \leftarrow \overline{A}$	Complement A
$F_{13} = x' + y$	$F \leftarrow \overline{A} \vee B$	
$F_{14} = (xy)'$	$F \leftarrow \overline{A \wedge B}$	NAND
$F_{15} = 1$	$F \leftarrow$ all 1's	Set to all 1's

Figure One stage of logic circuit.



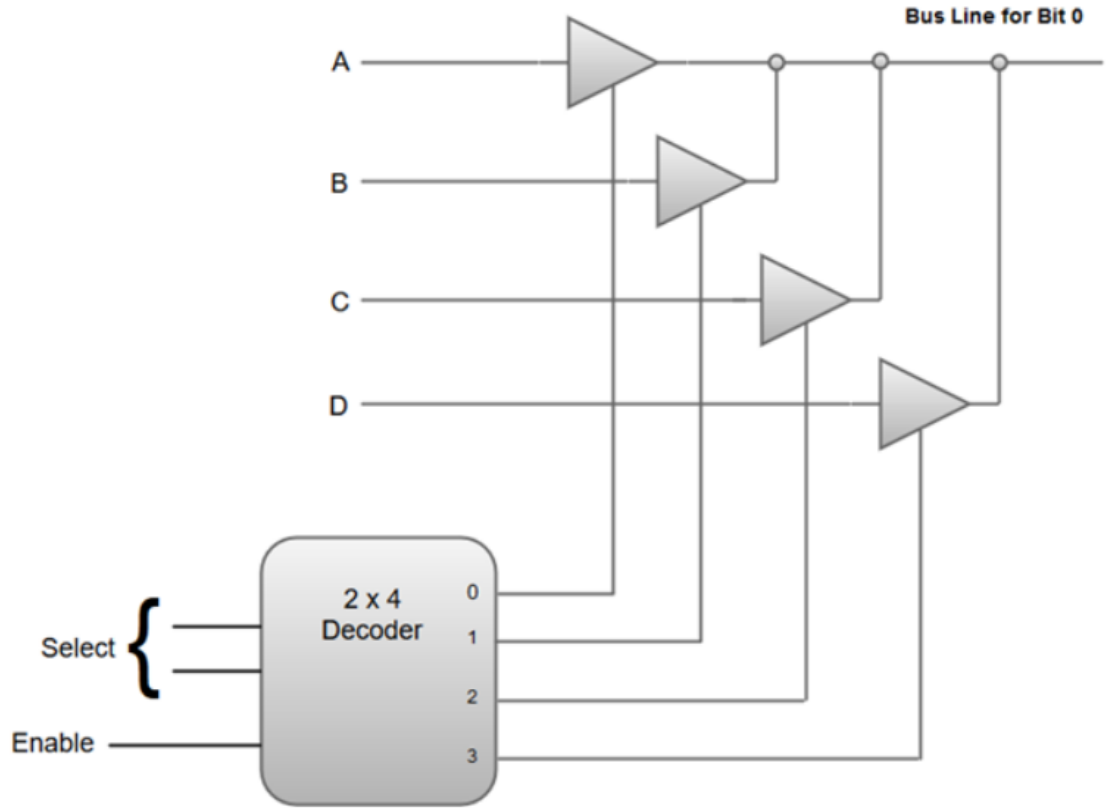
(a) Logic diagram

S_1	S_0	Output	Operation
0	0	$E = A \wedge B$	AND
0	1	$E = A \vee B$	OR
1	0	$E = A \oplus B$	XOR
1	1	$E = \overline{A}$	Complement

(b) Function table

Answer 12:

Bus line with three state buffer:



(consider this diagram as one complete system & you have to make seven more system that is equal to total 8 system and that will together form the required diagram)

Answer 13:

Computer instructions are a set of machine language instructions that a particular processor understands and executes. A computer performs tasks on the basis of the instruction provided.
There are three types of instructions:

Memory reference instructions: These instructions access or modify data stored in memory23. Examples are LOAD, STORE, MOVE, PUSH, etc.

(Opcode = 000 through 110)

Register reference instructions: These instructions perform arithmetic, logical, or shift operations on data in processor registers. Examples are ADD, SUB, AND, OR, NOT, etc.

(Opcode = 111, I = 0)

Input-output instructions: These instructions transfer data between the processor and external devices. Examples are IN, OUT, etc.

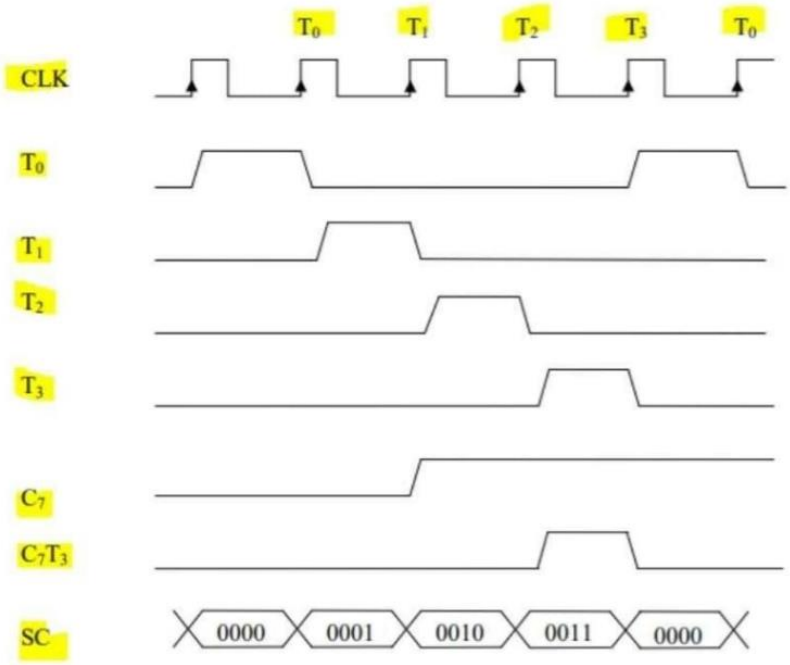
(Opcode = 111, I = 1)

Answer 14:

S.No.	Hardwired Control Unit	Microprogrammed Control Unit
1.	The hardwired control unit induces the control signals required for the processor.	The microprogrammed control unit induces the control signals through microinstructions.
2.	Hardwired control unit is quicker than a microprogrammed control unit.	Microprogrammed control unit is slower than a hardwired control unit.
3.	It is hard to modify.	It is easy to modify.
4.	It is more expensive as compared to the microprogrammed control unit.	It is affordable as compared to the hardwired control unit.
5.	It faces difficulty in managing the complex instructions because the design of the circuit is also complex.	It can easily manage complex instructions.
6.	It can use limited instructions.	It can generate control signals for many instructions.

Answer 15:

The Timing diagram is given below as per your requiement diagram:



Long Answer:

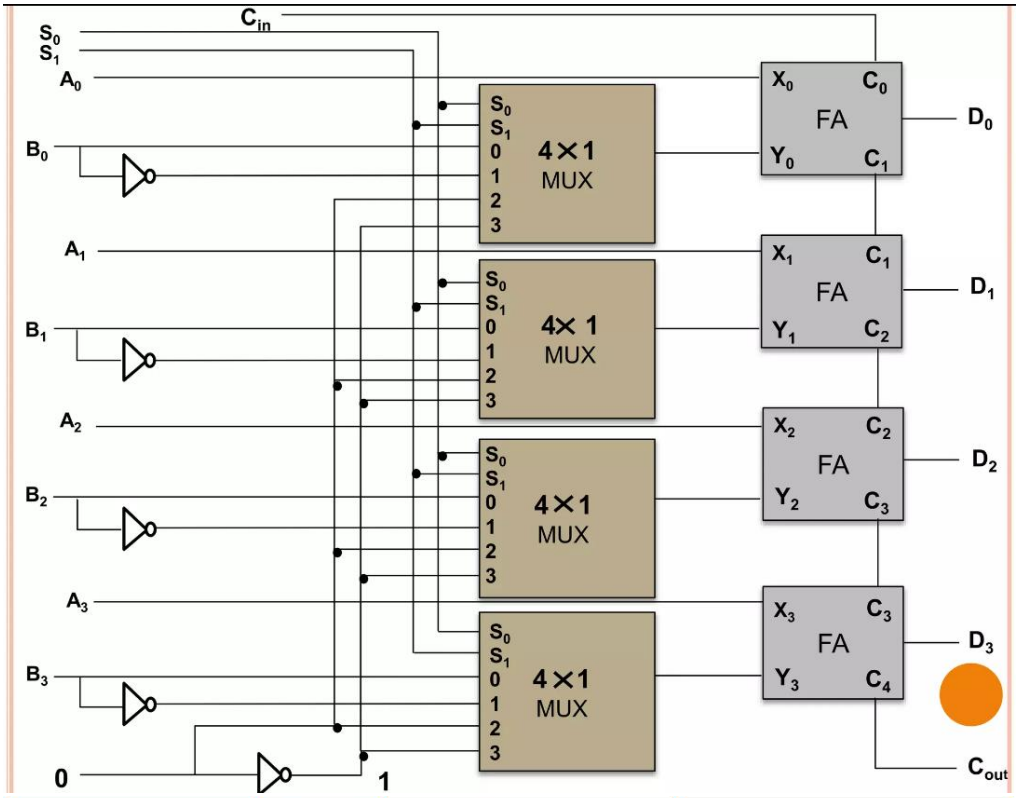
Answer 1:

A microoperation is an elementary operation performed with the data stored in Registers.

There are four categories of the most common microoperation:

1. Register transfer microoperation: transfer binary information from one register to another
2. Arithmetic microoperation: perform arithmetic operations on numeric data stored in registers.
3. Logic microoperation: perform bit manipulation operations on non-numeric data stored in registers.
4. Shift microoperation: perform shift operations on data stored in registers.

Implementation of Arithmetic Circuit:



Select			Input	Output	Micro operation
S_1	S_0	C_{in}	Y	$D = A + Y + C_{in}$	
0	0	0	B	D = A + B	Add
0	0	1	B	D = A + B + 1	Add with Carry
0	1	0	$\overline{B}$	$D = A + \overline{B}$	Subtract with Borrow
0	1	1	$\overline{B}$	$D = A + \overline{B} + 1$	Subtract
1	0	0	0	D = A	Transfer A
1	0	1	0	D = A + 1	Increment A
1	1	0	1	D = A – 1	Decrement A
1	1	1	1	D = A	Transfer A

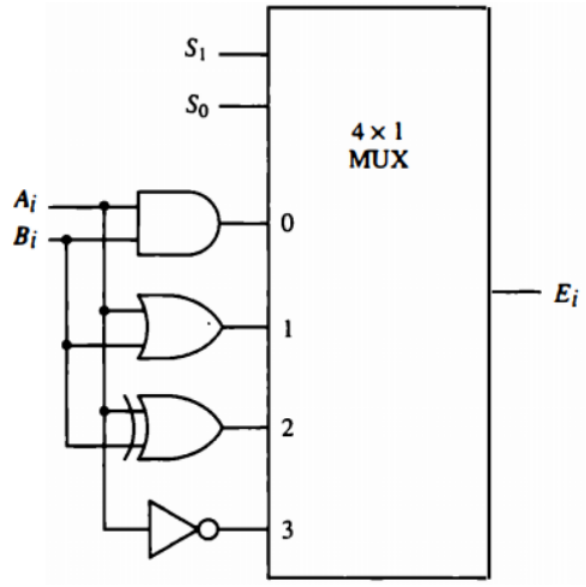
The arithmetic micro-operations can be implemented in one composite arithmetic circuit. It has four full adder circuits that constitute the four bit adder and four multiplexers for choosing different operations. There are two four bit inputs A and B and a four bit output D. The four inputs from A go directly to the X inputs of the binary adder. Each of the four inputs from the B are connected to the data input of the multiplexer.

The output of the binary adder is calculated from the following arithmetic sum:

$$D = A + Y + C_{in}$$

Implementation of logic circuit:

Figure One stage of logic circuit.



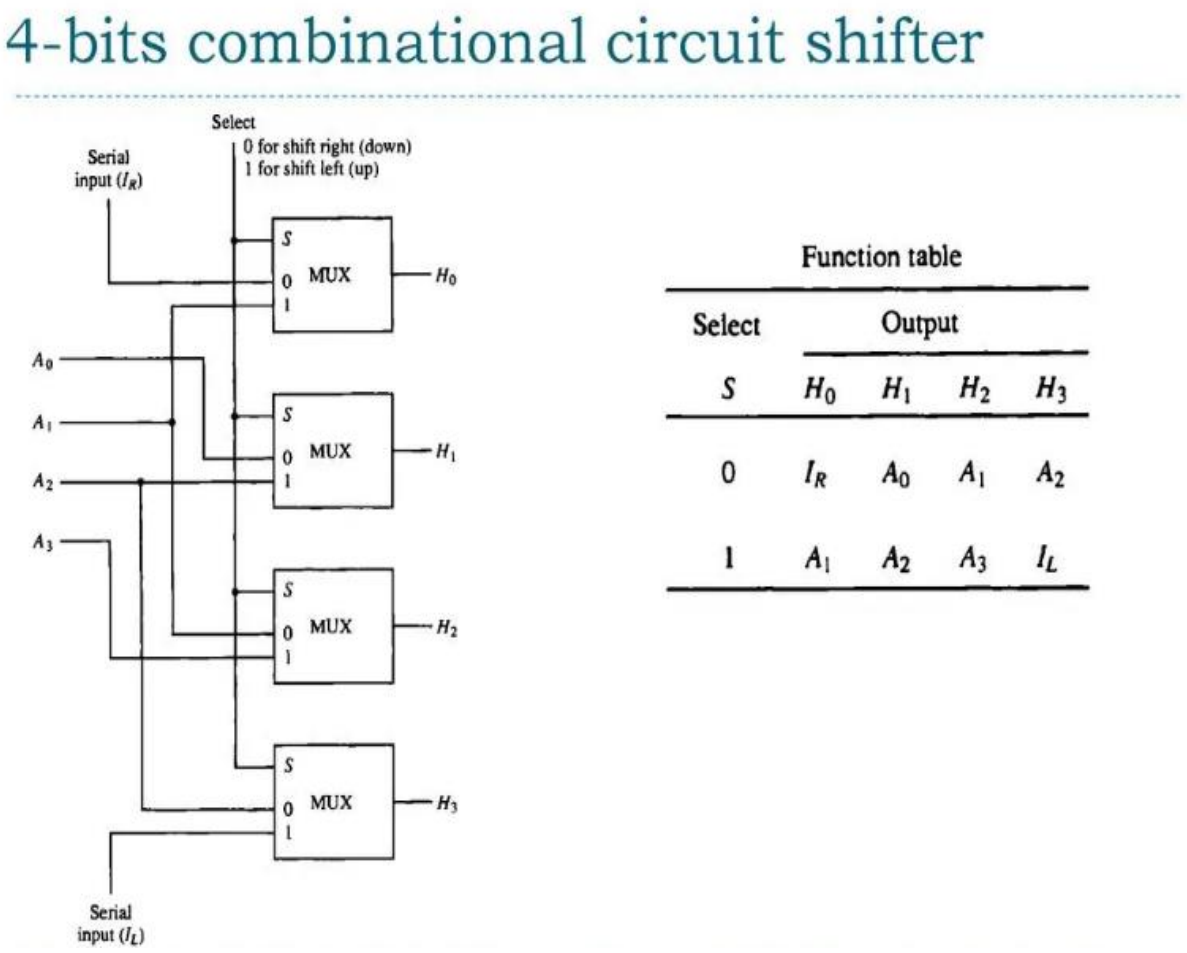
(a) Logic diagram

S_1	S_0	Output	Operation
0	0	$E = A \wedge B$	AND
0	1	$E = A \vee B$	OR
1	0	$E = A \oplus B$	XOR
1	1	$E = \overline{A}$	Complement

(b) Function table

- The hardware implementation of logic rnicrooperations requires that logic gates be inserted for each bit or pair of bits in the registers to perform the required logic function.
- Although there are 16 logic microoperations, most computers use only four-AND, OR, XOR (exclusive-OR), and complement from which all others can be derived.
- Figure below shows one stage of a circuit that generates the four basic logic microoperations.
- It consists of four gates and a multiplexer. Each of the four logic operations is generated through a gate that performs the required logic.
- The outputs of the gates are applied to the data inputs of the multiplexer. The two selection inputs S1 and S0 choose one of the data inputs of the multiplexer and direct its value to the output.
- The diagram shows one typical stage with subscript i. For a logic circuit with n bits, the diagram must be repeated n times for i = 0, 1, 2, ... , n - 1.
- The selection variables are applied to all stages. The function table in Fig. below lists the logic microoperations obtained for each combination of the selection variables.

Implementation of Shift micro-operation:



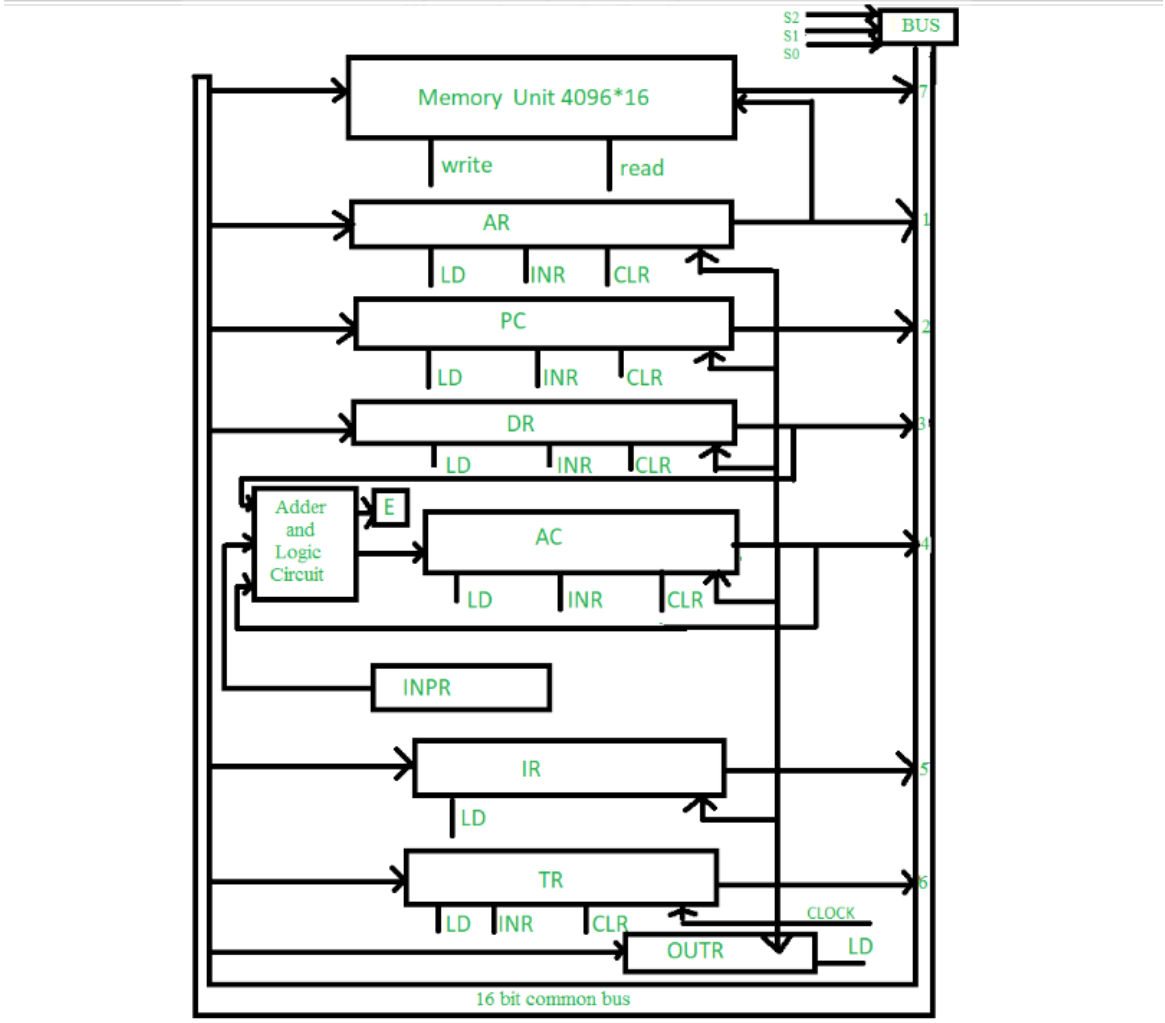
Function table

Select	Output			
S	H_0	H_1	H_2	H_3
0	I_R	A_0	A_1	A_2
1	A_1	A_2	A_3	I_L

A 4-bit shifter has four data inputs A_3 to A_0 four data output H_3 to H_0. There are 4 serial inputs, one four shift left(I_L) and the other for shift right(I_R). When the selection input S = 0, the input data are shifted right (down in the diagram). When S = 1, the input data are shifted left (up in the diagram).

Answer 2:

Bus is a subsystem that is used to transfer data and other information between devices. Various devices in computer such as memory, CPU, I/O etc. are communicate with each other through buses.



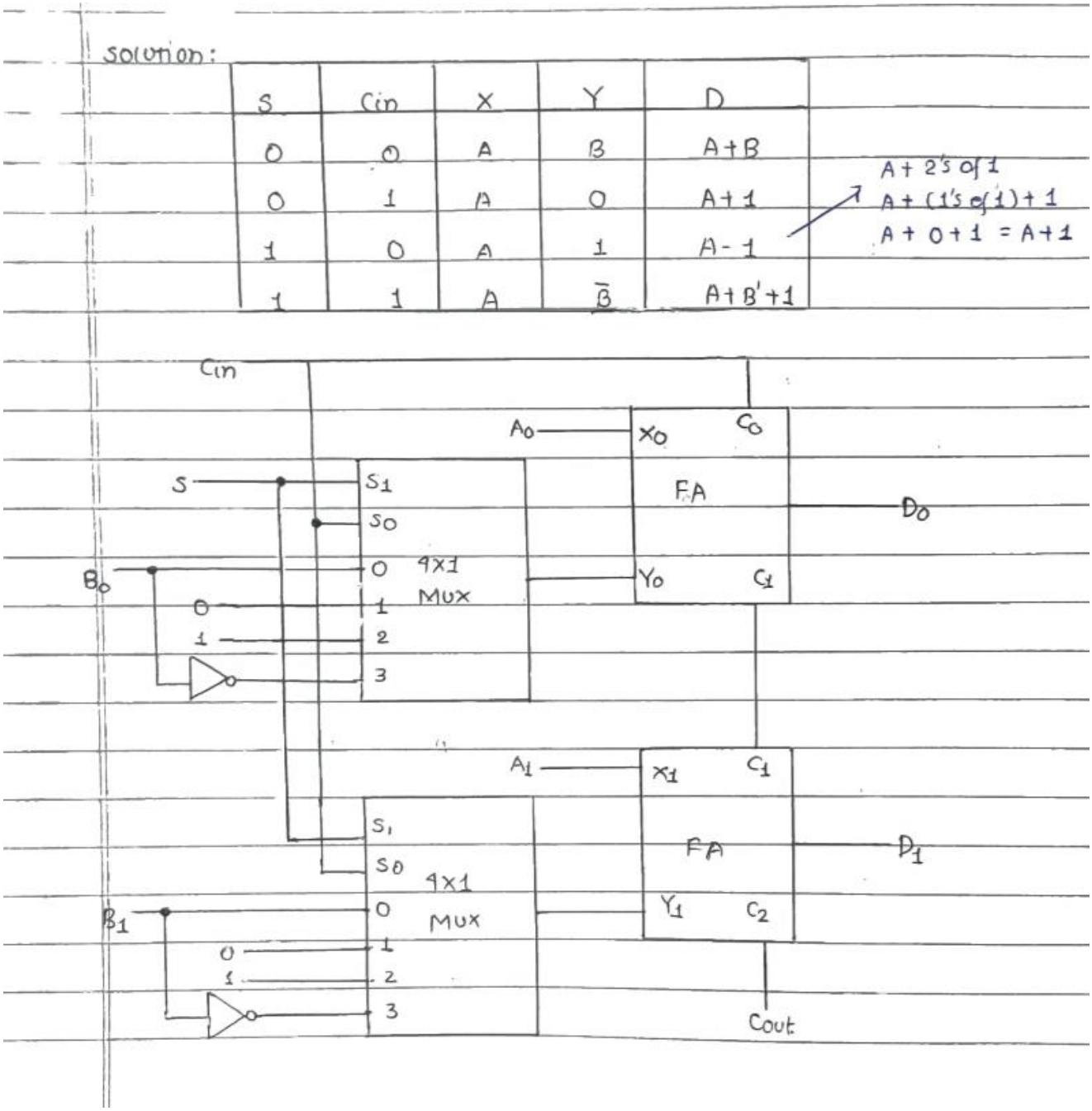
Connections:

The outputs of all the registers except the OUTR (output register) are connected to the common bus. The output selected depends upon the binary value of variables S2, S1 and S0. The lines from common bus are connected to the inputs of the registers and memory. A register receives the information from the bus when its LD (load) input is activated while in case of memory the Write input must be enabled to receive the information. The contents of memory are placed onto the bus when its Read input is activated.

Various Registers:

4 registers DR, AC, IR and TR have 16 bits and 2 registers AR and PC have 12 bits. The INPR and OUTR have 8 bits each. The INPR receives character from input device and delivers it to the AC while the OUTR receives character from AC and transfers it to the output device. 5 registers have 3 control inputs LD (load), INR (increment) and CLR (clear). These types of registers are similar to a binary counter.

Answer 3:



Answer 4:

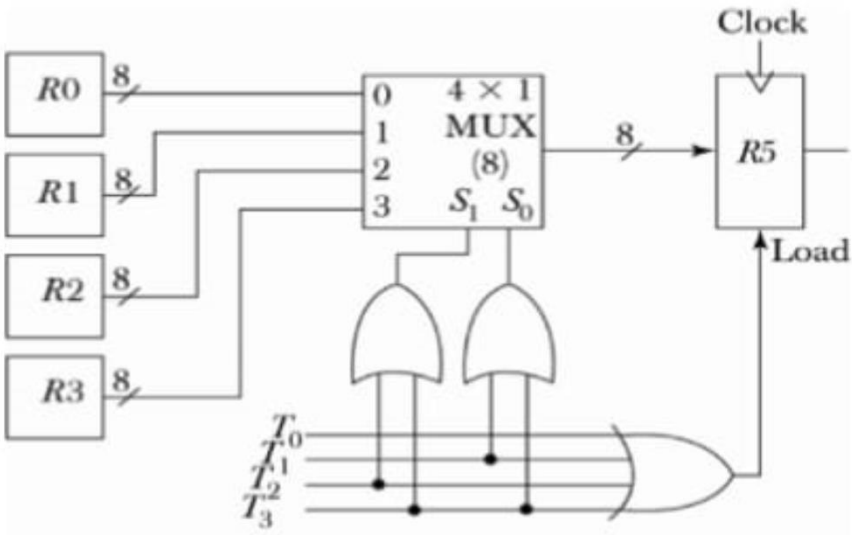
Solution:

The truth table for the outputs:

T0	T1	T2	T3	S1	S0	Load
0	0	0	0	x	x	0
1	0	0	0	0	0	1
0	1	0	0	0	1	1
0	0	1	0	1	0	1
0	0	0	1	1	1	1

Use the "sum of product" technique to get the logic expressions for the selection lines and the load signal:

```
S1 = T2 + T3
S0 = T1 + T3
Load = T0 + T1 + T2 + T3
```



Answer 5:

The instruction cycle is the cycle that the CPU follows to process instructions. It consists of the following phases:

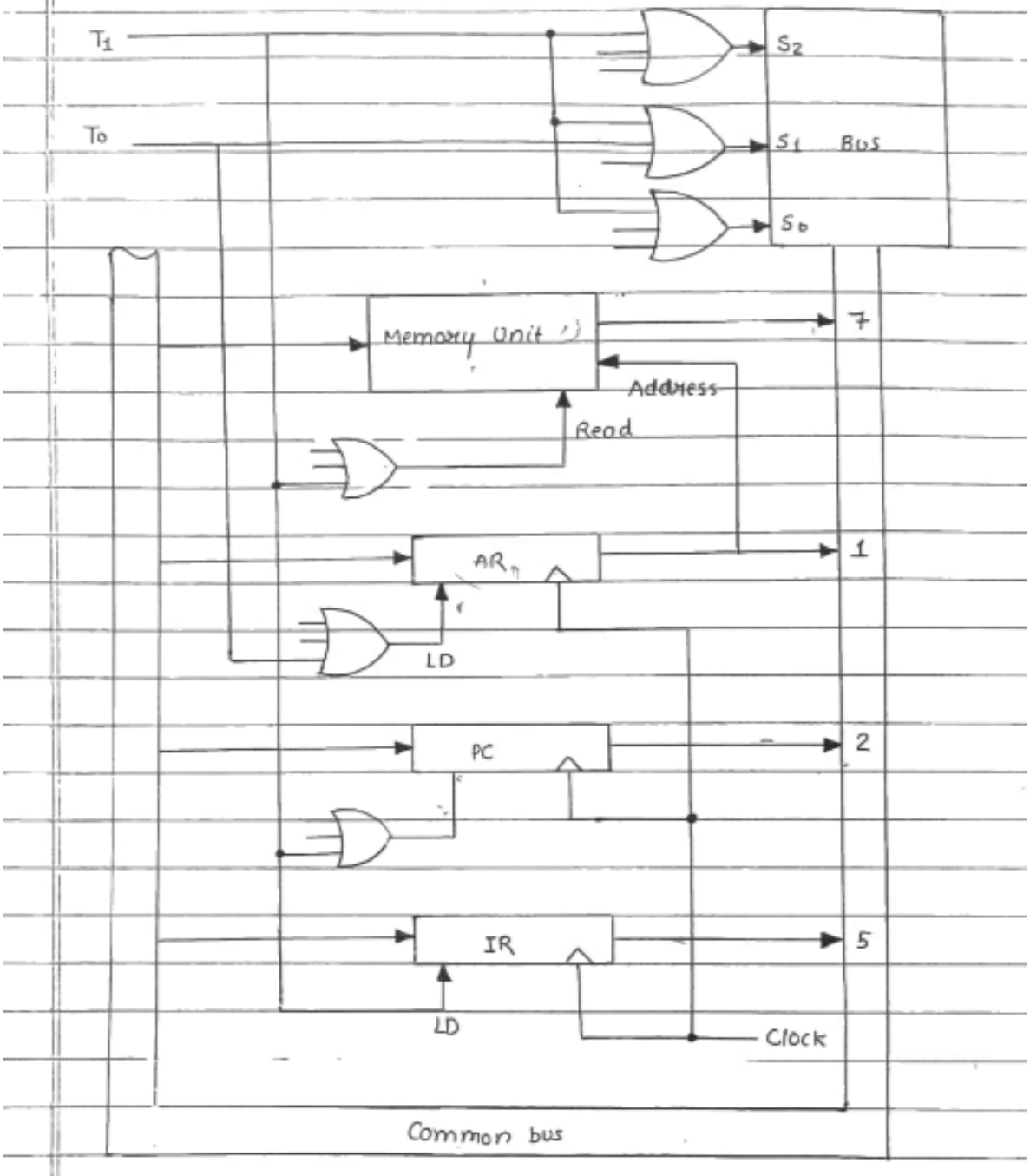
- Fetch the instruction from memory.
- Decode the instruction.
- Read the effective address from memory if the instruction has an indirect address.
- Execute the instruction.
- Memory access or registry write-back if needed.

Microoperation associated with fetch and decode phases can be represented with following register transfer statements.

→ T0 : AR ← PC

→ T1 : IR ← M[AR], PC ← PC + 1

→ T2 : D0 .... D7 ← Decode IR(12-14), AR ← IR(0-11), I ← IR(15)



[Implementation of Register Transfer Instruction for fetch cycle]

Register Transfer statements for fetch cycle are :

T_0 : $AR \leftarrow PC$

T_1 : $IR \leftarrow M[AR]$, $PC \leftarrow PC + 1$

- To achieve the following connections :
  - → Place the content of PC onto the bus by making the bus selection inputs $S_2S_1S_0 = 010$
  - → Transfer the content of the bus to AR by enabling the LD (Load) input of AR.

- T_1 provide the following connection :
  - → Enable the read input of memory
  - → Place the content of memory onto the bus by making $S_2S_1S_0 = 111$
  - → Transfer the content of the bus to IR by enabling the LD (Load) input of AR.
  - → Increment PC by enabling the INR input of AC.

Answer 6:

Symbol	Operation decoder	Symbolic description
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR]$, $E \leftarrow C_{out}$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[AR] \leftarrow PC$, $PC \leftarrow AR + 1$
ISZ	D_6	$M[AR] \leftarrow M[AR] + 1$, If $M[AR] + 1 = 0$ then $PC \leftarrow PC + 1$

BUN: Branch Unconditionally

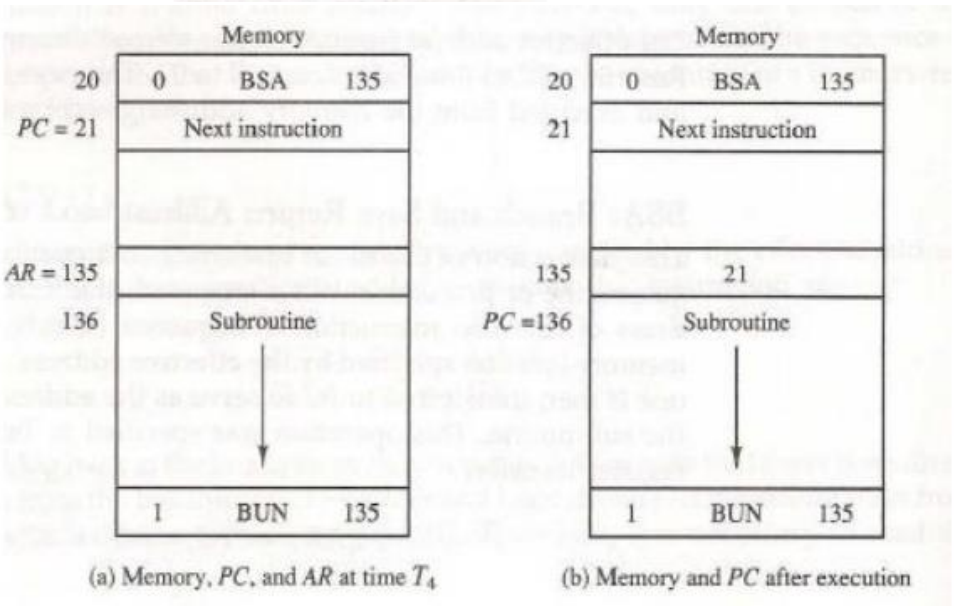
- This instruction transfers the program to the instruction specified by the effective address.
- The BUN instruction allows the programmer to specify an instruction out of sequence and we say that the program branches (or jumps) unconditionally.
- The instruction is executed with one microoperation:

D_4T_4: $PC \leftarrow AR$, $SC \leftarrow 0$

BSA: Branch and Save Return Address

- This instruction is useful for branching to a portion of the program called a subroutine or procedure.
- When executed, the BSA instruction stores the address of the next instruction in sequence (which is available in PC) into a memory location specified by the effective address.
- The effective address plus one is then transferred to *PC* to serve as the address of the first instruction in the subroutine.
- This operation was specified with the following register transfer:

D_5T_4: $M[AR] \leftarrow PC, \quad AR \leftarrow AR + 1$
D_5T_5: $PC \leftarrow AR, \quad SC \leftarrow 0$



(a) Memory, *PC*, and *AR* at time T_4 (b) Memory and *PC* after execution

Answer 7:

	Instruction code	Hexadecimal code	Symbol	Symbol Description	Description
a)	0001 0000 0010 0100	1024	ADD	AC <- AC + M[AR], E <- C_{out}	Add memory word to AC
b)	1011 0001 0010 0100	B124	STA	M[AR] <- AC	Store content of memory into AC
c)	0111 1000 0000 0000	7800	CLA	AC <- 0	Clear AC
d)	0111 0000 1000 0000	7080	CIR	Ac <- shr AC, AC(15) <- E, E <- AC(0)	Circular Right AC and E
e)	1111 1000 0000 0000	F800	INP	AC(0-7) <- INPR, FGI <- 0	Input Character to AC