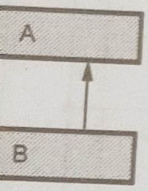
micro-operations that
operations on non
s.

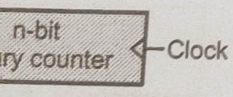
micro-operations that
ta stored in registers.

conditional control
fer statements with

that executes the



implements the
logic gates for
diagram for the
enable input.



on a common
nation between
ts of a set of
gister, through
one at a time.
e which is the
ion register for

The common bus scheme can be implemented in two ways -

- Using multiplexers
- Using tri-state bus buffers.

Q.9 Explain the implementation of common bus system using multiplexers.

Ans. : • The Fig. Q.9.1 shows the implementation of common bus system for four registers using multiplexers. Each register has four bits, numbered 0 through 3 and they are routed through multiplexers to the common bus. (See Fig. Q.9.1 on next page)

- Here, four multiplexers are used to select four bits of the source register. Each multiplexers has four input lines, two select lines and one output line. The four input lines of multiplexer 0 (MUX 0) are connected to the bit 0 outputs of four registers such that bit 0 of register is connected to input 0, bit 0 of register 1 is connected to input 1, bit 0 of register 2 is connected to input 2 and bit 0 of register 3 is connected to input 3. Similarly, inputs for MUX 1 are connected to bit 1 outputs, inputs for MUX 2 are connected to bit 2, and inputs for MUX 3 are connected to bit 3 outputs of registers 0 through 3.
- To avoid the complexity of the diagram, only input connections for MUX 3 are physically shown. To show other connections labels are used. Two same labels have connection between them. For example, bit-2 ($R_0 - B_2$) output of register 0 is connected to the input 0 of MUX 2.
- The two selection lines S_1 and S_0 are connected to the selection inputs of all four multiplexers. These lines choose the four bits of one register and transfer them into the four-line common bus through OUT lines.
- When S_1 S_0 = 00, the input 0 of all four multiplexers are selected and applied to the outputs to transfer them on the common bus. As a result a common bus receives the contents of register 0. Since the outputs of register 0 are connected to the input 0 of the multiplexers. Similarly, the content of register 1 are transferred on the common bus when S_1 S_0 = 01.

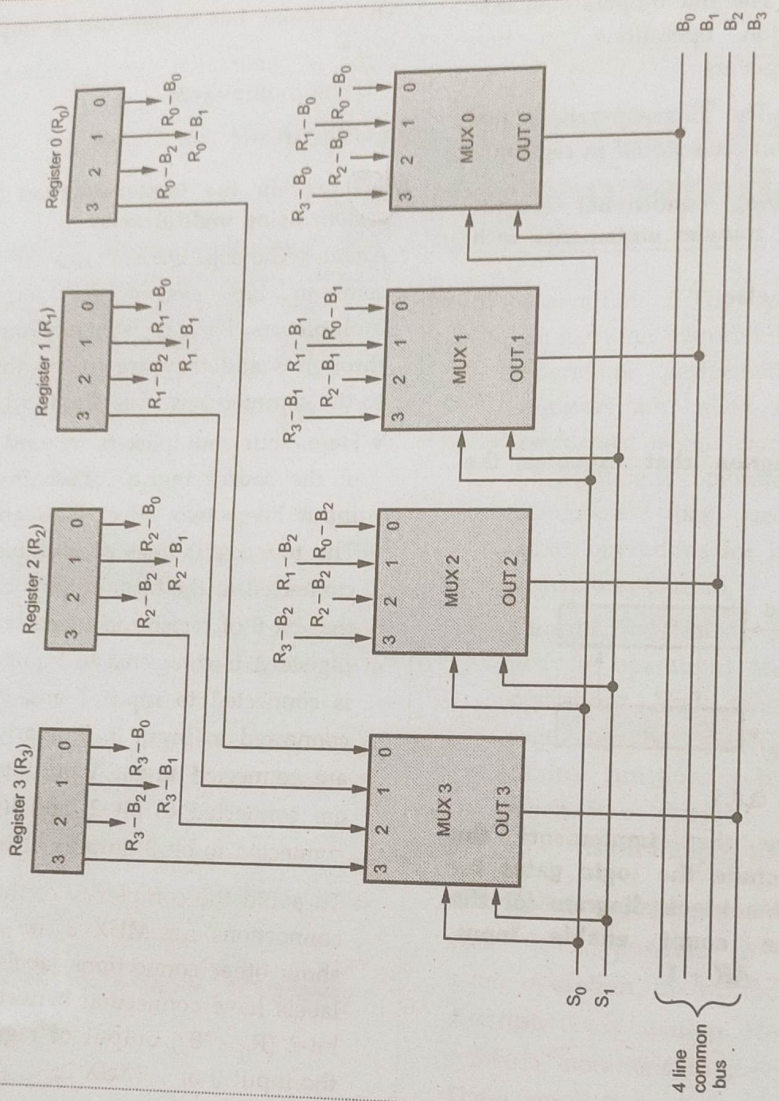


Fig. Q.9.1 Common bus system using multiplexers

- The Table Q.9.1 shows the register selection according to the status of S_1 S_0 lines.

S_1	S_0	Selected register
0	0	Register 0
0	1	Register 1
1	0	Register 2
1	1	Register 3

Table Q.9.1 Selection table

- In general, common bus system will have n multiplexers and n-line common bus where n represents the number of bits in each register and k number of inputs to each multiplexer where k represents the total number of registers. Therefore, the size of each multiplexer will be k × 1.

Q.10 Design a circuit for parallel load operation into one of the four 4-bit registers from a bus. Mention clearly control/selection bits and selection logic. Assume JK flip-flops.

☞ **[JNTU : May-09, Marks 16]**

Ans. : The Fig. Q.10.1 shows the circuit for the desired operation. It consists of four 4-bit registers with JK flip-flops. Load inputs for registers are derived using 2:4 decoder which decides the destination register in which data from the common bus is to be loaded. The Table Q.10.1 shows the function table for the circuit.

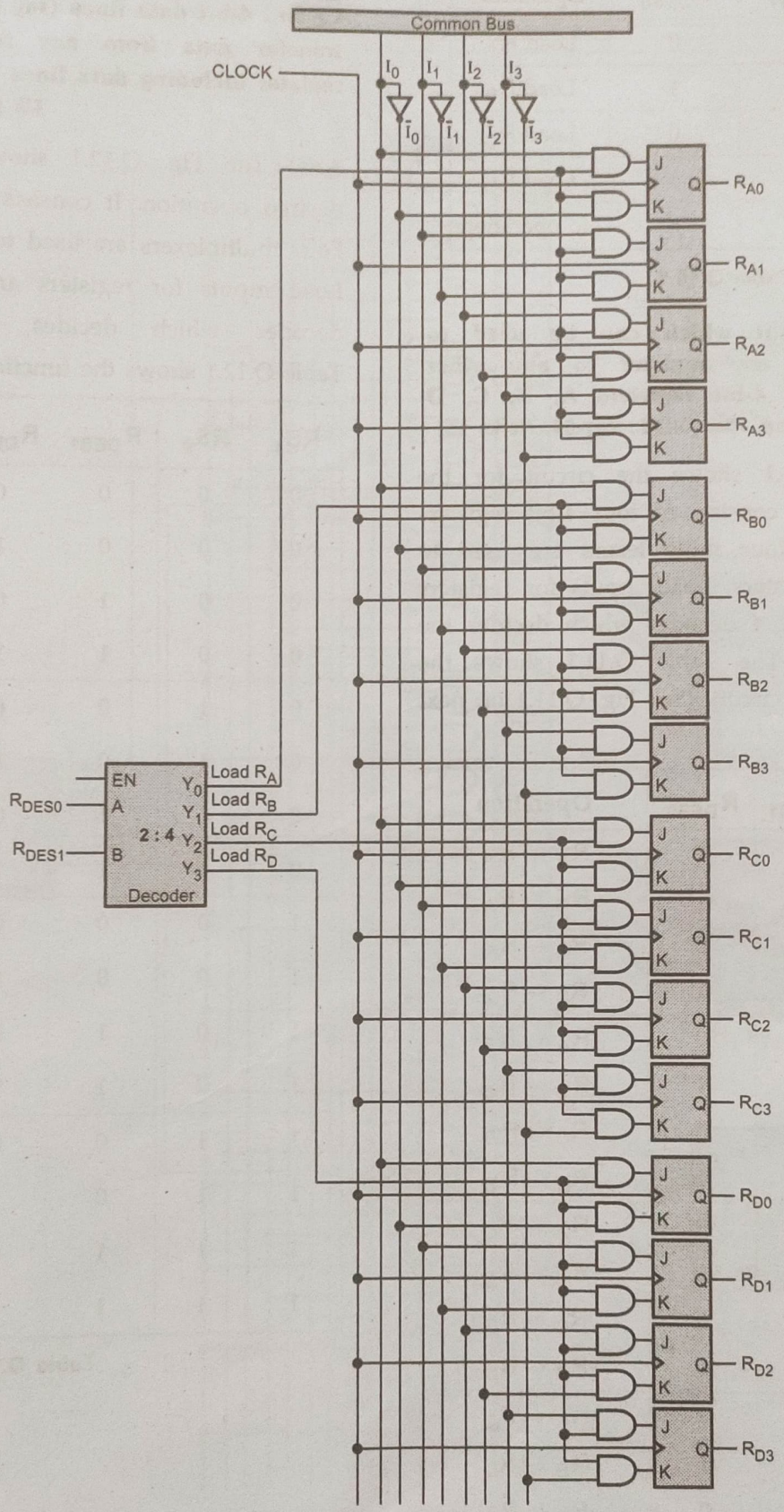

Common Bus
CLOCK
I_0 I_1 I_2 I_3
$\bar{I}_0$ $\bar{I}_1$ $\bar{I}_2$ $\bar{I}_3$
J Q K
R_{A0} R_{A1} R_{A2} R_{A3}
R_{B0} R_{B1} R_{B2} R_{B3}
R_{C0} R_{C1} R_{C2} R_{C3}
R_{D0} R_{D1} R_{D2} R_{D3}
EN
A
B
2 : 4 Decoder
Y_0 Y_1 Y_2 Y_3
R_{DES0}
R_{DES1}
Load R_A
Load R_B
Load R_C
Load R_D

Fig. Q.10.1

Output

```
cout<<"\n Enter the string";
cin>>name;
```


E	R_{DES1}	R_{DES0}	Operation
1	0	0	Load R_A
1	0	1	Load R_B
1	1	0	Load R_C
1	1	1	Load R_D
0	x	x	No operation

Table Q.10.1

Q.11 Design a circuit which can be used to transfer data from any register to any other register out of four 4-bit registers A, B, C, D which uses RS flip-flop. ☞ **[JNTU : May-09, Marks 16]**

Ans. : The Fig. Q.11.1 shows the circuit for the desired operation. It consists of four 4-bit registers with RS flip-flops. Four multiplexers are used to select the source registers. Load inputs for registers are derived using 2 : 4 decoder which decides the destination register. The Table Q.11.1 shows the function table for the circuit. (See Fig. Q.11.1 on next page)

RS_1	RS_0	R_{DES1}	R_{DES0}	Operation
0	0	0	0	$R_A \leftarrow R_A$
0	0	0	1	$R_B \leftarrow R_A$
0	0	1	0	$R_C \leftarrow R_A$
0	0	1	1	$R_D \leftarrow R_A$
0	1	0	0	$R_A \leftarrow R_B$
0	1	0	1	$R_B \leftarrow R_B$
0	1	1	0	$R_C \leftarrow R_B$
0	1	1	1	$R_D \leftarrow R_B$
1	0	0	0	$R_A \leftarrow R_C$
1	0	0	1	$R_B \leftarrow R_C$
1	0	1	0	$R_C \leftarrow R_C$
1	0	1	1	$R_D \leftarrow R_C$
1	1	0	0	$R_A \leftarrow R_D$
1	1	0	1	$R_B \leftarrow R_D$
1	1	1	0	$R_C \leftarrow R_D$
1	1	1	1	$R_D \leftarrow R_D$

Table Q.11.1

Q.12 There exists three 4-bit registers (A, B and C) and 4-bit data lines (say D). Design a circuit to transfer data from any register to any other register including data lines and vice-versa.

☞ **[JNTU : May-09, Marks 16]**

Ans. : The Fig. Q.12.1 shows the circuit for the desired operation. It consists of three 4-bit registers. Four multiplexers are used to select the data source. Load inputs for registers are derived using 2 : 4 decoder which decides the destination. The Table Q.12.1 shows the function table for the circuit.

RS_1	RS_0	R_{DES1}	R_{DES0}	Operation
0	0	0	0	$R_A \leftarrow R_A$
0	0	0	1	$R_B \leftarrow R_A$
0	0	1	0	$R_C \leftarrow R_A$
0	0	1	1	$D \leftarrow R_A$
0	1	0	0	$R_A \leftarrow R_B$
0	1	0	1	$R_B \leftarrow R_B$
0	1	1	0	$R_C \leftarrow R_B$
0	1	1	1	$D \leftarrow R_B$
1	0	0	0	$R_A \leftarrow R_C$
1	0	0	1	$R_B \leftarrow R_C$
1	0	1	0	$R_C \leftarrow R_C$
1	0	1	1	$D \leftarrow R_C$
1	1	0	0	$R_A \leftarrow D$
1	1	0	1	$R_B \leftarrow D$
1	1	1	0	$R_C \leftarrow D$
1	1	1	1	$D \leftarrow D$

Table Q.12.1

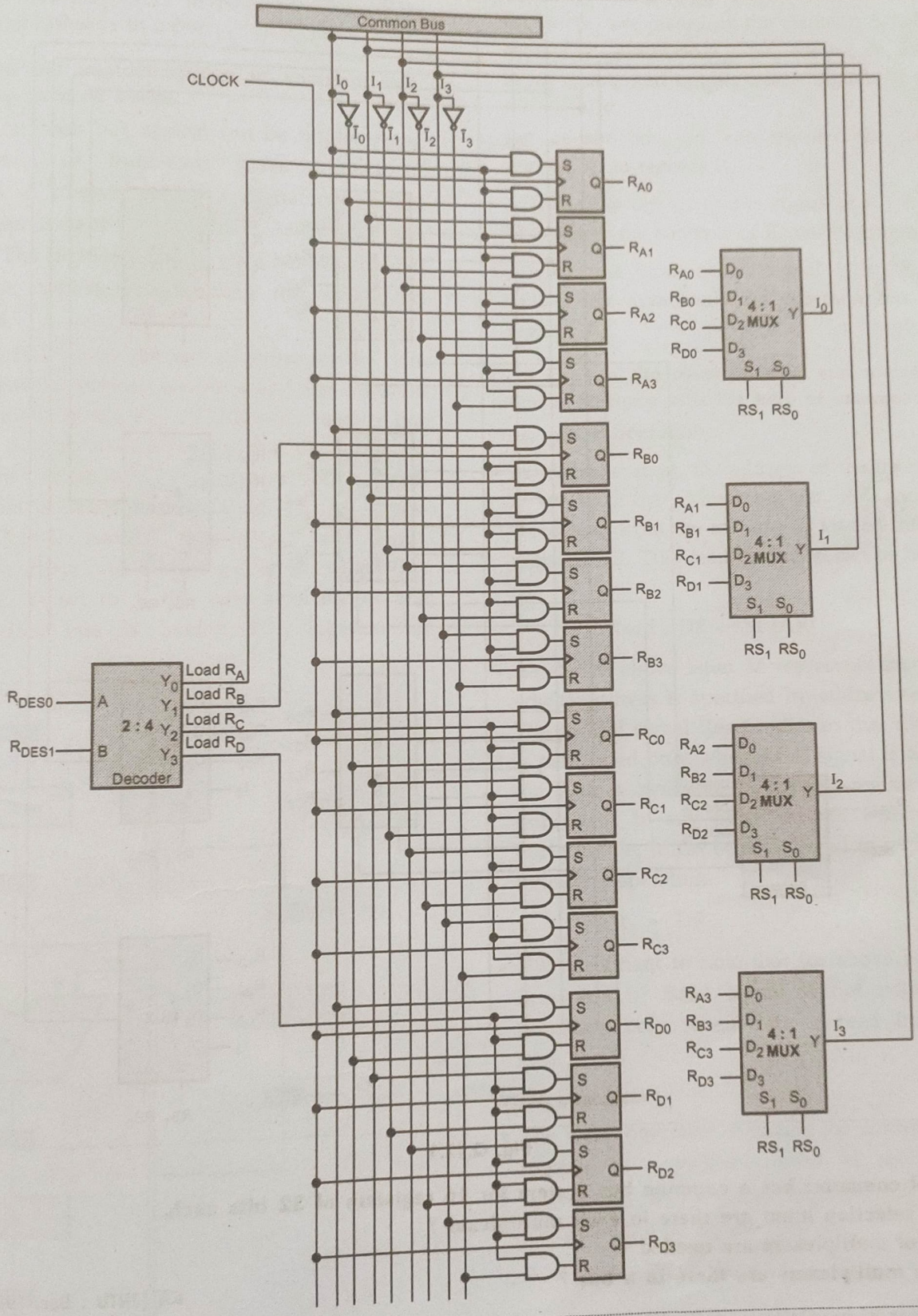

Common Bus
CLOCK
I_0 I_1 I_2 I_3
$\bar{I}_0$ $\bar{I}_1$ $\bar{I}_2$ $\bar{I}_3$
S Q R
R_{A0} R_{A1} R_{A2} R_{A3}
R_{B0} R_{B1} R_{B2} R_{B3}
R_{C0} R_{C1} R_{C2} R_{C3}
R_{D0} R_{D1} R_{D2} R_{D3}
R_{A0} R_{B0} R_{C0} R_{D0} D_0 D_1 D_2 D_3 4 : 1 MUX Y I_0 S_1 S_0 RS_1 RS_0
R_{A1} R_{B1} R_{C1} R_{D1} I_1
R_{A2} R_{B2} R_{C2} R_{D2} I_2
R_{A3} R_{B3} R_{C3} R_{D3} I_3
R_{DES0} R_{DES1} A B 2 : 4 Decoder Y_0 Y_1 Y_2 Y_3
Load R_A
Load R_B
Load R_C
Load R_D

Fig. Q.11.1

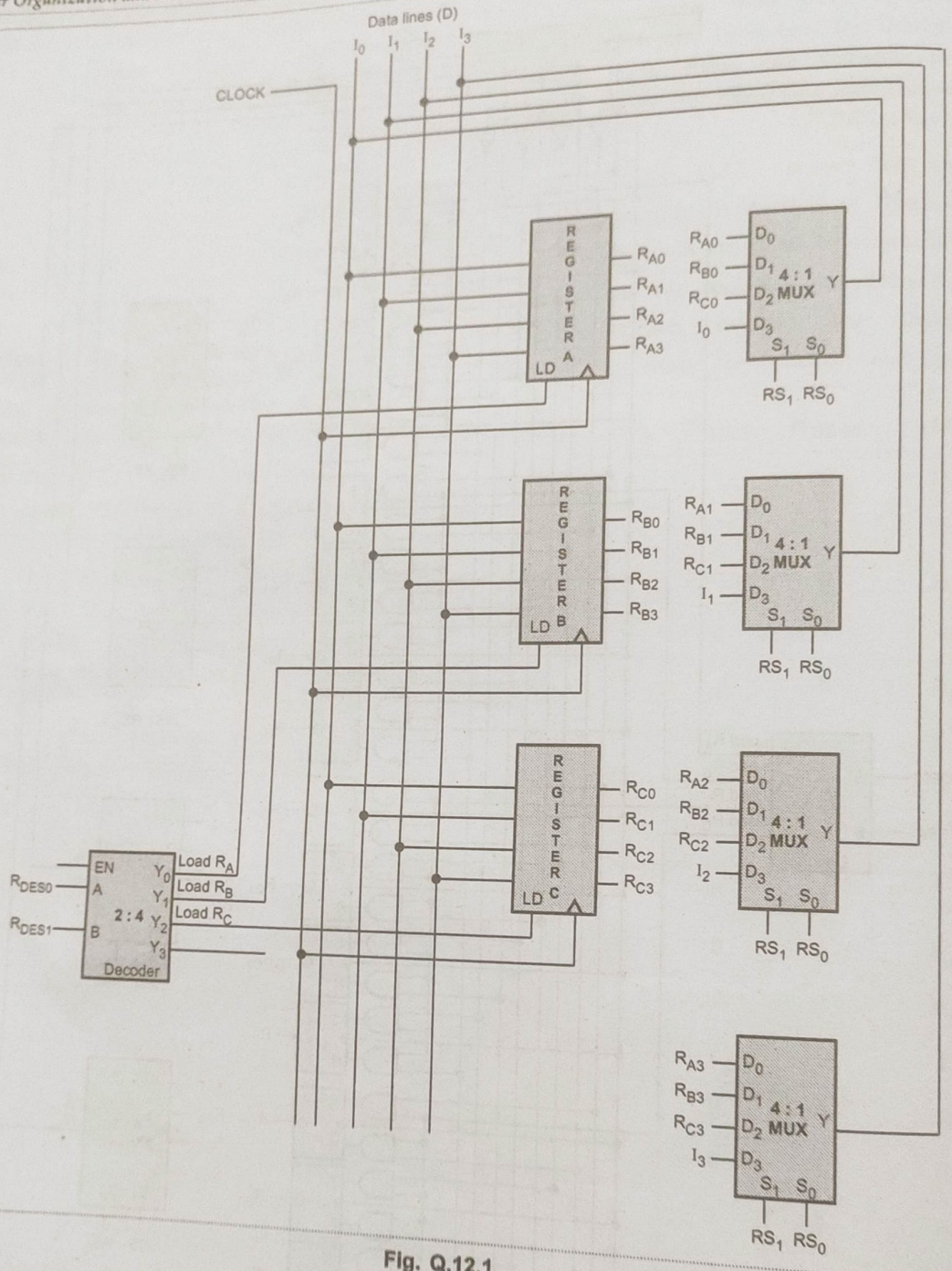


Fig. Q.12.1

Q.13 A digital computer has a common bus system for 16 registers of 32 bits each.
i) How many selection input are there in each multiplexer ?
ii) What size of multiplexers are needed ?
iii) How many multiplexers are there in a bus ?

☞ **[JNTU : Dec.-19, Marks 10]**

Ans. : i) Number of inputs to each multiplexer is the total number of registers, i.e. 16. With 16 inputs multiplexer must have 4 selection inputs.

ii) Size of the multiplexer will be 16 : 1.

Output

Enter the string : Computer
You have entered : Computer

before the last printf statement we
in for visiting

- The initialization of structure can be using scanf statement or by using direct assignment of values to structure variables. For example
struct stud stud1={1,"ABC",99.99};
struct stud stud2={2,"XYZ",80};
Usually the structure declaration should be do

iii) Total number of bits in each register gives the number of multiplexers to be used. Thus there are 32 multiplexers in a bus.

Q.14 Explain the implementation of common bus system using tri-state buffer.

Ans. : • A common bus system can be implemented using **tri-state** or **three-state gates** instead of multiplexers. A tri-state gate is a digital circuit that can have three output states : HIGH, LOW and high impedance. The high impedance state behaves like an open circuit, which means that the output is disconnected.

- The Fig. Q.14.1 shows the bus structure for the data transfer between various registers and the common bus. As shown in the Fig. Q.14.1, each register has input and output tri-state buffers and these tri-state buffers are controlled by corresponding control signals. This is illustrated in Fig. Q.14.1. As shown in Fig. Q.14.1, control signals $R_{i\,in}$ and $R_{i\,out}$ controls the input and output gating of register R_i. When $R_{i\,in}$ is set to 1, the data available on the common data bus is loaded into register R_i.

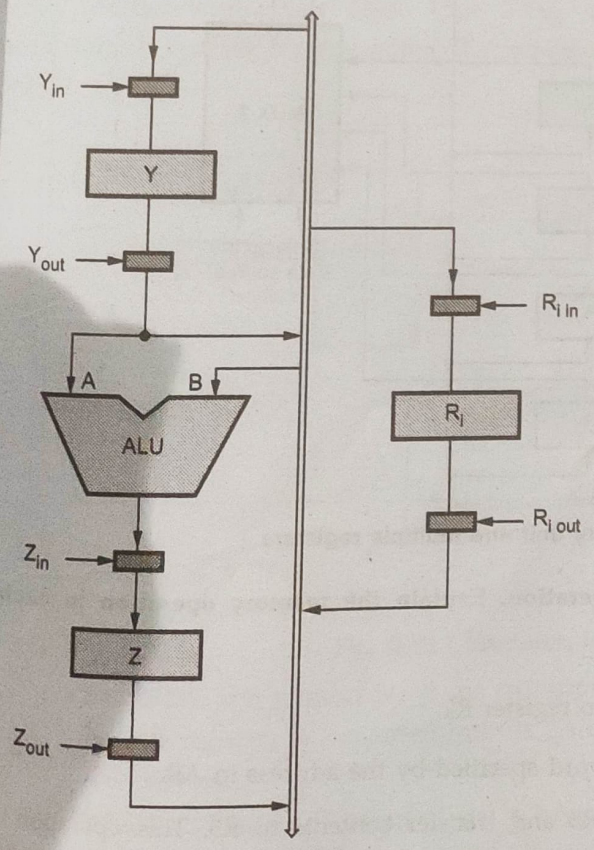


Fig. Q.14.1 Input and output tri-state buffers for the register

Similarly, when $R_{i\,out}$ is set to 1, the contents of register R_i are placed on the common data bus. The signals $R_{i\,in}$ and $R_{i\,out}$ are commonly known as input enable and output enable signals of registers, respectively.

Let us see how we can transfer the data from register R_1 to register R_2.

- Activate the output enable signal of R_1, $R_{1\,out} = 1$. This places the contents of R_1 on the common bus.
- Activate the input enable signal of R_2, $R_{2\,out} = 1$. This loads data from the common bus into the register R_2.

Q.15 Explain the memory read and memory write micro-operations with the help of examples.

Ans. : Read operation :

- In read operation, the address of the memory word is specified by address register, AR and the data word read from the memory is loaded into the data register, DR. This read operation can be represented as,

$$\textbf{Read} : DR \leftarrow M\,[AR]$$

- As shown above letter M represents memory word whose address is specified by address register, AR. The control signal **Read** indicates that the operation is performed only when **Read** signal is active. Once the data is available in the DR register it can be transferred to any other register within the CPU. For example, it can be transferred to register, R_2 by following operation.

$$R_2 \leftarrow DR$$

- It is important to note that for above operation the activation of **Read** signal is not necessary; it is necessary only when data is read from memory unit.

Write operation :

- The write operation transfers the contents of a data register to a memory word M selected by the address in the address register, AR. The write operation can be represented as,

$$\textbf{Write} : M\,[AR] \leftarrow DR$$

- The control signal **Write** indicates that the operation is performed only when **Write** signal is active. When it is necessary to transfer data from any

2.2 : Arithmetic Micro-Operations

Q.18 Explain various arithmetic micro-operations.

Ans. : [JNTU : Nov-04, 05, Dec-14, Marks 7]

Operation	Representation	Description
Add	$R_3 \leftarrow R_1 + R_2$	Contents of R_1 and R_2 are added and result is transferred to R_3.
Subtract	$R_3 \leftarrow R_1 - R_2$	Content of R_2 are subtracted from content of R1 and result is transferred to R_3.
1's Complement	$\overline{R}_1$	Complement the content of R_1.
2's complement	$\overline{R}_1 + 1$	Complement the contents of R_1 and add 1 in it.
2's complement subtraction	$R_3 \leftarrow R_1 + \overline{R}_2 + 1$	Add R_1 and the 2's complement of R_2.
Increment	$R_1 \leftarrow R_1 + 1$	Increment the contents of R_1 by one.
Decrement	$R_1 \leftarrow R_1 - 1$	Decrement the contents of R_1 by one.

Table Q.18.1 Arithmetic micro-operations

Q.19 Draw and explain the hardware to implement integer addition and subtraction.

[JNTU : May-17 Marks 10]

Ans. : Addition/Subtraction Logic Unit : Fig. Q.19.1 shows hardware to implement integer addition and subtraction. It consists of n-bit adder, 2's complement circuit, overflow detector logic circuit and AVF (overflow flag).

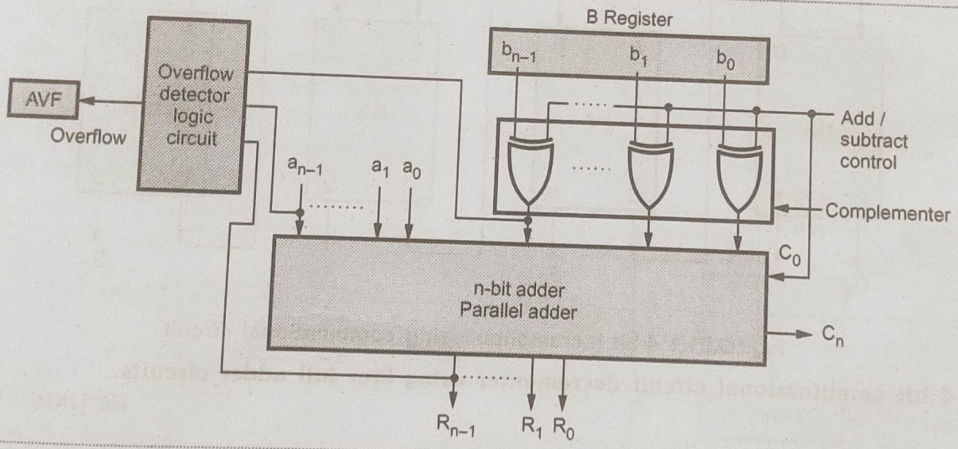


Fig. Q.19.1 Hardware for integer addition and subtraction

- Number a and number b are the two inputs for n-bit adder. For subtraction, the subtrahend (number from B register) is converted into its 2's complement form by making Add/Subtract control signal to the logic one.
- When Add/Subtract control signal is one, all bits of number b are complemented and carry zero (C_0) is set to one. Therefore n-bit adder gives result as $R = a + \overline{b} + 1$, where $\overline{b} + 1$ represents 2's complement of number b.

Q.20 Indicate how an overflow is detected ?

Ans. : 1. Overflow can occur only when adding two numbers that have the same sign.

2. The carry bit from the MSB position is not a sufficient indicator of overflow when adding signed numbers.

3. When both operands a and b have the same sign, an overflow occurs when the sign of result does not agree with the signs of a and b. The logical expression to detect overflow can be given as

$$\text{Overflow} = a_{n-1}\, b_{n-1}\, \overline{R_{n-1}} + \overline{a_{n-1}}\, \overline{b_{n-1}}\, R_{n-1}$$

where a_{n-1} = MSB of number a

b_{n-1} = MSB of number b

R_{n-1} = MSB of the result

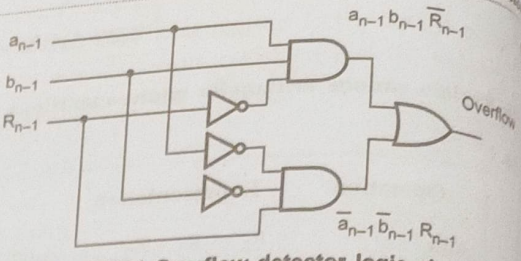


Fig. Q.20.1 Overflow detector logic circuit

Q.21 Draw and explain the 4-bit incrementer using combinational circuit.

Ans. : Fig. Q.21.1 shows a combinational circuit to implement increment micro-operation. Here, one of the inputs to the least significant half-adder (HA) is connected to logic 1 and the other input is connected to the least significant bit of the number to be increment. The output carry from lower order half adder is connected to the one of the inputs of the next higher order half adder. Such circuit gets the four bit number from bits A_0 through A_3, adds one to it, and generates the incremented output in S_0 through S_3. If number to be incremented is 1111 the output carry C_4 is generated and we get 0000 at S_0 through S_3.

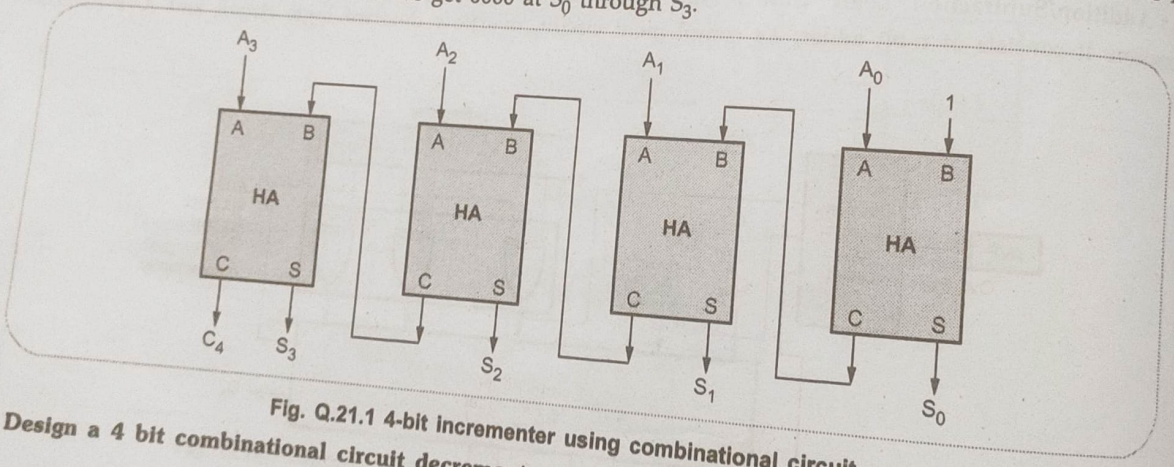


Fig. Q.21.1 4-bit incrementer using combinational circuit

Q.22 Design a 4 bit combinational circuit decrementer using four full adder circuits.

[JNTU : Dec.-19, Marks 5]

Ans. : Let us consider A register is a 4-bit register.

A – 1 = A + 2's complement of 1.

2's complement of 1 = 1's complement of 1(0001 + 1) = 1110 + 1 = 1111

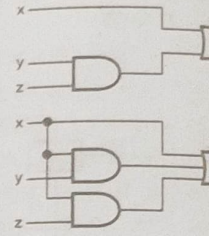
This operation can be implemented using logic circuit shown in Fig. Q.22.1.

Q.23 Draw the block diagram for the hardware that implements the following statements :

i) x + yz : AR ← AR + BR

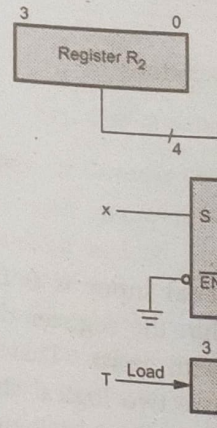
ii) x + xy + xz : AR ← AR + 1.

Ans. : i) The Fig. Q implements statemer statement x + xy + xz



Q.24 Consider the statements for two 4 xT : $R_1 \leftarrow R_1 + R_2$, Every time that var of R_2 is added to the content of R_2 is Draw a diagran implementation of diagram for the tw and a quadruple 2: the input to R_1. In variables x and multiplexer and the

Ans. :



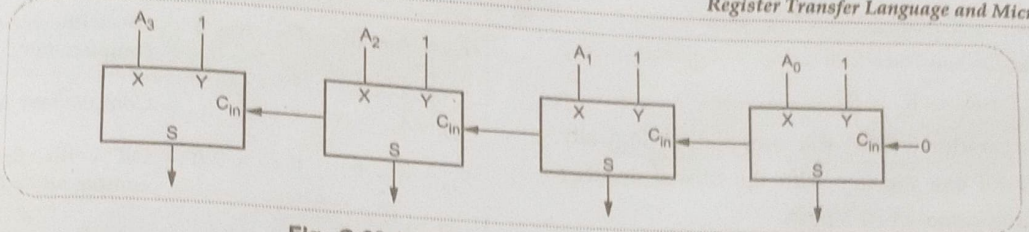


Fig. Q.22.1 4-bit combinational decrementor

Ans. : i) The Fig. Q.23.1 shows the hardware that implements statement $x + yz : AR \leftarrow AR + BR$ and statement $x + xy + xz : AR \leftarrow AR + 1$.

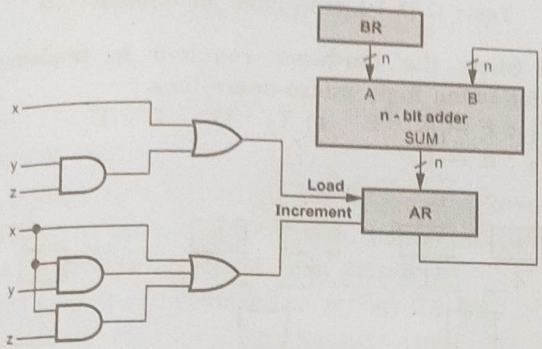


Fig. Q.23.1

Q.24 Consider the following register transfer statements for two 4-bit registers R_1 and R_2
$xT : R_1 \leftarrow R_1 + R_2$, $\bar{x}T = R_1 \leftarrow R_2$
Every time that variable $T = 1$, either the contents of R_2 is added to the content of R_1 if $x = 1$ or the content of R_2 is transferred to R_1 if $x = 0$.
Draw a diagram showing the hardware implementation of the two statements. Use block diagram for the two 4-bit register, a 4-bit adder and a quadruple 2:1 line multiplexer that selects the input to R_1. In the diagram show the control variables x and T, select the inputs of the multiplexer and the load input of register R_1.

Ans. :

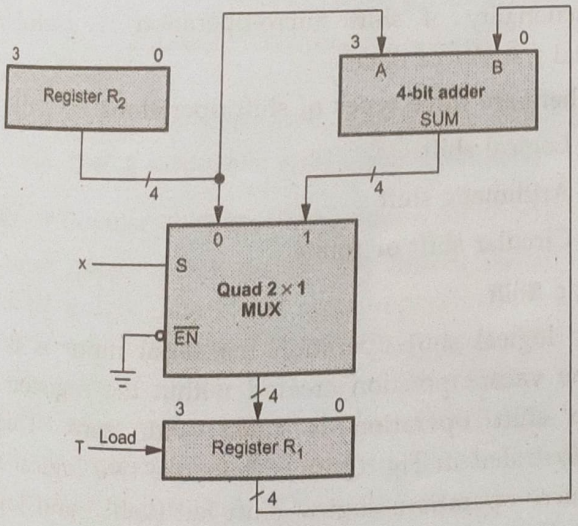


Fig. Q.24.1

Q.25 Show the hardware implementation for following statement.
$\bar{X}\,Y\,T_0 + T_1 + \bar{X}\,Y\,T_2 : A \leftarrow A + B$

Ans. :

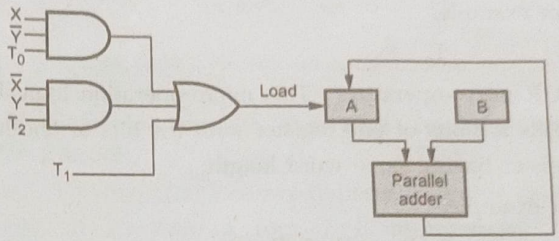


Fig. Q.25.1

Q.26 Define RTL and Micro-operation. Write RTL statement for following :
If (p = 1) then R1 = R1 + R1 else if (p = 0) then R1 = R1 + R2. Also draw the block diagram for the same. ☞ [JNTU : June-06]

Ans. : $P : R1 \leftarrow R1 + R1$
$\bar{P} : R1 \leftarrow R1 + R2$

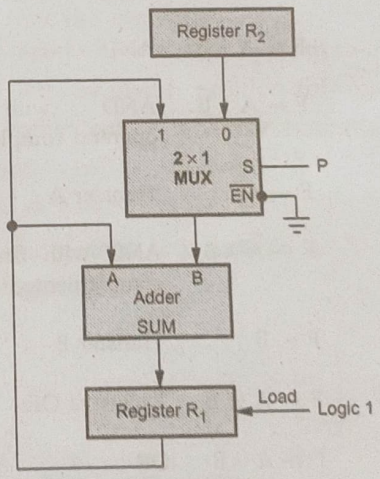


Fig. Q.26.1

2.3 : Logic Micro Operations

Q.27 Explain basic logic micro-operations.

Ans. : AND micro-operation : This micro-operation logically ANDs the bits of one register with the bits of another register having same word length.

For example,

$$T : R_1 \leftarrow R_1 \wedge R_2$$

OR micro-operation : This micro-operation logically ORs the bits of one register with the bits of another register having same word length.

For example,

$$T_1 : R_1 \leftarrow R_1 \vee R_2$$

Complement micro-operation : This micro-operation complements all bits in the register.

For example,

$$T_2 : \overline{R}_1$$

XOR micro-operation : This micro-operation logically XORs the bits of one register with the bits of another register having same word length.

For example,

$$T : R_1 \leftarrow R_1 \oplus R_2$$

Q.28 List various logic micro-operations.

[JNTU : Nov.-04, 05, Feb-10, July-14, Marks 7]

Ans. :

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

Table Q.28.1 List of logic micro-operations

Q.29 Show the hardware required to implement the following logic micro-operations :

a) $T_1 : F \leftarrow A \wedge B$ b) $T_2 : G \leftarrow C \vee D$

c) $T_3 : E \leftarrow \overline{E}$.

Ans. :

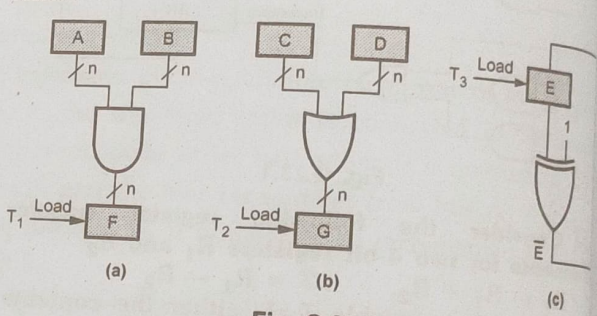


Fig. Q.29.1

2.4 : Shift Micro Operations

Q.30 Explain various shift micro operations.

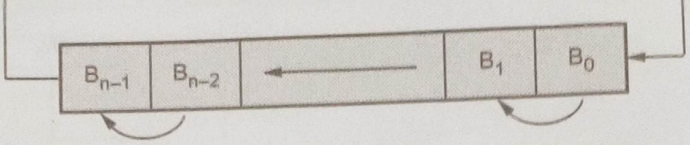
OR Explain the difference between arithmetic and logical right shift.

Ans. : • Shift micro-operations are used to shift the contents of a register to the left or the right. This functionality of shift micro-operation is useful for serial transfer of data.

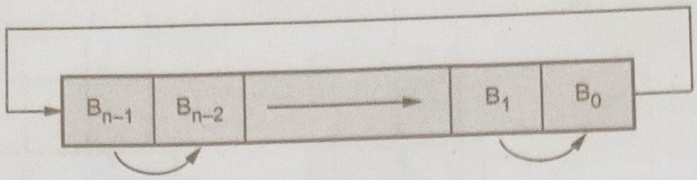
- There are three types of shift operations :
  - Logical shift
  - Arithmetic shift
  - Circular shift or rotate

Logic Shift

- In logical shift operation the serial input is 0 i.e. the vacant position created within the register due to shift operation is filled with zero. This is illustrated in Fig. Q.30.1. There are two logical shift micro-operations logical shift left (LshL) and logical shift right (LshR). For example,



(a) Rotate left



(b) Rotate right

Fig. Q.30.3 Rotate micro-operations

Q.31 Draw and explain the 4-bit combinational shifter circuit.

Ans. : The Fig. Q.31.1 shows the combinational shifter circuit. It consists of four 2 : 1 multiplexers. The shifter circuit has four data inputs, A_0 through A_3, and four data outputs, Z_0 through Z_3. There are two serial inputs, one for shift left and the other for shift right. The select input is used to select shift right or shift left operation. When S = 1, shift left operation is performed and when S = 0 shift right operation is performed. (See Fig. Q.31.1 on next page)

Q.32 Show that the statement :
$A \leftarrow A + A$ symbolizes a shift left micro-operation.

Ans. : Shift left by 1-bit position of any register contents is equivalent to multiply contents by 2, i.e. adding contents itself (A * 2 = A + A).

For example, A = 06

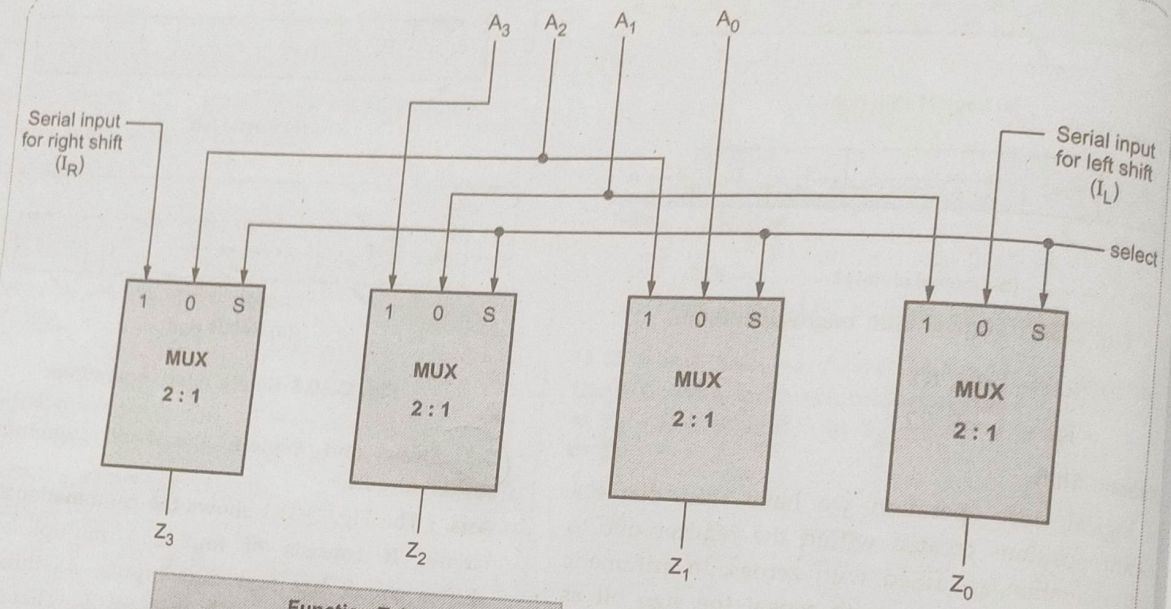
Results are same, thus it is proved at A + A = Left shift by 1-bit position

Add operation

		1	1			← Carry
	A	0	1	1	0	(6)
+	A	0	1	1	0	+ (6)
A+A		1	1	0	0	12

Left shift operation

0	1	1	0	Initial contents of A
1	1	0	0	Contents of A shifted by 1-bit



Function Table	
Select	Output
S	Z_3 Z_2 Z_1 Z_0
0	A_2 A_1 A_0 I_L
1	I_R A_3 A_2 A_1

Note :

S = 0 Left shift operation
S = 1 Right shift operation

Fig. Q.31.1 4-bit combinational shifter circuit

Q.33 An 8-bit register A has one input q. The register operation is described symbolically as $Y : A_1 \leftarrow q$, $A_{i+1} \leftarrow A_i$, $i = 1, 2, 3 \ldots 7$.

Ans. : The register operation described symbolically can be drawn as follows :

The Fig. Q.33.1 shows that it is shift left register with serial input q and control input Y.

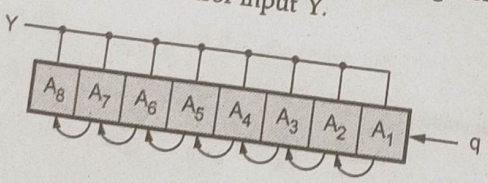


Fig. Q.33.1

2.5 : Arithmetic Logic Shift Unit

Q.34 Briefly explain arithmetic logic shift unit with help of a functional table ?

Ans. : Fig. Q.34.1 shows one stage of an arithmetic logic shift unit. The subscript i designates a typical stage. Inputs A_i and B_i are applied to both the arithmetic and logic units. Inputs S_1 and S_0 are used to select a particular micro-operation. A 4 : 1 multiplexer at the output chooses between an arithmetic output in Y_i and a logic output in Z_i. The data in the multiplexer are selected with inputs S_1 and S_2. The other two data inputs to the multiplexer receive inputs A_{i-1} for shift left operation and A_{i+1} for shift right operation. It is important to note that the diagram shown in Fig. Q.34.1 shows just one typical stage. The circuit shown in Fig. Q.34.1 must be repeated n times for an n-bit ALU. For multistage ALU, the output carry C_{i+1} of a previous arithmetic stage must be connected to the input carry C_i of the next stage in sequence.

Ans. : The Table Q.5.1 summarize the various registers and their functions.

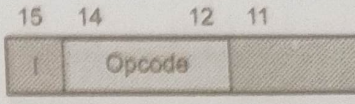
Register	Symbol used	Number of bits	Function
Data Register or Memory Data Register	DR or MDR	16	Holds the data read from memory or to be written in the memory.
Address Register or Memory Address register	AR or MAR	12	Holds the address for the memory.
Accumulator Register	AC	16	Holds the operand for the ALU and can be used as a general-purpose registers.
Instruction Register	IR	16	Holds the instruction code.
Program Counter	PC	12	Holds the address of the next instruction to be executed.
Temporary Register	TR	16	Holds the temporary data.
Input Register	INPR	8	Holds the data read from input device.
Output Register	OUTR	8	Holds the data to be sent to the output device.

Table Q.5.1 Basic computer registers

3.3 : Computer Instructions

Ans. : The Fig. Q.7.1 sho
memory reference in
reference instruction has

- 1-Bit (I) specifies a
  indirect
- 3-Bits (Opcode) spec
- 12-Bits (Address) sp



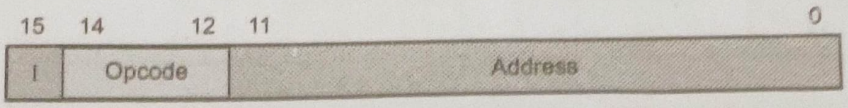
I = 0 : Direct addressing m
I = 1 : Indirect addressing
Opcode : It can have value fr
since there are 7 m

Fig. Q.7.1 Memory

Q.8 List various m with their functional

Ans. :

Mnemonic	
AND	Logic specif AC ←
ADD	Adds locat AC
LDA	Loac loca AC
STA	Stor spe M
BUN	Un spe PC



I = 0 : Direct addressing mode
I = 1 : Indirect addressing mode
Opcode : It can have value from 000 through 110
since there are 7 memory reference instructions

Fig. Q.7.1 Memory reference instruction format

Q.8 List various memory reference instructions with their functionality. ☞ **[JNTU : Nov.-16, Marks 5]**

Ans. :

Mnemonic	Description
AND	Logically ANDs the contents of specified memory location and AC AC ← AC × M [AR]
ADD	Adds the contents of specified memory location with AC AC ← AC + M [AR]
LDA	Loads the contents of specified memory location in AC AC ← M [AR]
STA	Store the contents of AC in the specified memory location M [AR] ← AC
BUN	Unconditionally branches to the specified address PC ← AR
BSA	Unconditionally branches and saves return address M [AR] ← PC, PC ← AR + 1
ISZ	Increments the contents of specified memory location and skip if it is zero. DR ← M [AR], DR ← DR + 1 M [AR] ← DR if (DR = 0) then PC ← PC + 1

he format of register

instruction specifies an
ormed with AC register.
d to access memory and
ecify an operation or a

0
er operation

instruction format

erence instructions

register reference

ption
of AC register
ister and E bit
ter and E bit
ontents of AC
ntents of AC
tents of AC
= 0.

ons

t/output

ructions,
nemory
uctions
12-bits
test to

Q.12 List various input/output instructions with their functionality. [JNTU : Dec.-15, Marks 2]

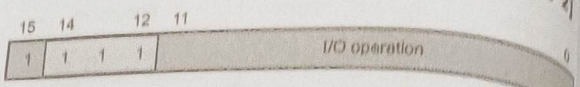


Fig. Q.11.1 Input-output instruction format

Ans. :

Mnemonic	Description
INP	Load a character in AC register from input port.
OUT	Send a character to output port from AC register.
SKI	Skip next instruction if input flag = 1.
SKO	Skip next instruction if output flag = 1.
ION	Enable interrupt, IEN ← 1.
IOF	Disable interrupt, IEN ← 0.

Table Q.12.1 Input-output instructions

3.4 : Timing and Control

Q.13 What is hardwired and microcontrolled control ?

Ans. : There are two different approaches to implement control unit : **Hardwired control** and **microprogrammed control**. In the hardwired control, the control units use fixed logic circuits to interpret instructions and generate control signals from them. Here, the fixed logic circuit block includes combinational circuit that generates the required control outputs for decoding and encoding functions. By separating the decoding and encoding functions.

In the microprogrammed organization, the control information is stored in a control memory. The control memory is programmed to initiate the required sequence of micro-operation to execute the processor instruction.

Q.14 Draw the block diagram of control unit of basic computer. Explain in detail with control timing diagrams.

OR Explain showing a basic block diagram, how the control unit of a CPU can be designed using hardwired control.

Ans. : The Fig. Q.14.1 (a) shows the block diagram of control unit which uses hardwired approach. It consists of decoders, a sequence counter and combinational logic circuitry.

The Fig. Q.14.1 (a) shows how the instruction in the Instruction Register (IR) is used to generate control signals. The instruction in the instruction register is divided into three fields : I field, operation code field and bits 0 through 11. The 3 : 8 decoder decodes the operation code bits and outputs of this decoder are used as an inputs for combinational logic circuitry. The four bit sequence counter counts from 0000 through 1111, i.e. (0 through 15 in decimal). These four bits are decoded by 4 : 16 decoder to generate timing signals from 0 through 15. The I bit from IR is also connected to the combinational control circuitry. Therefore, combinational control circuitry takes the input from 3 : 18 decoder, 4 : 16 decoder, I bit and bits 0 through 11 to generate control signals.

Along with the clock input Sequence Counter (SC) has two other inputs : Clear and increment. Initially, sequence counter is cleared by activating clear input. Once the SC is cleared, clear signal is negated and SC is incremented with every positive clock transition to

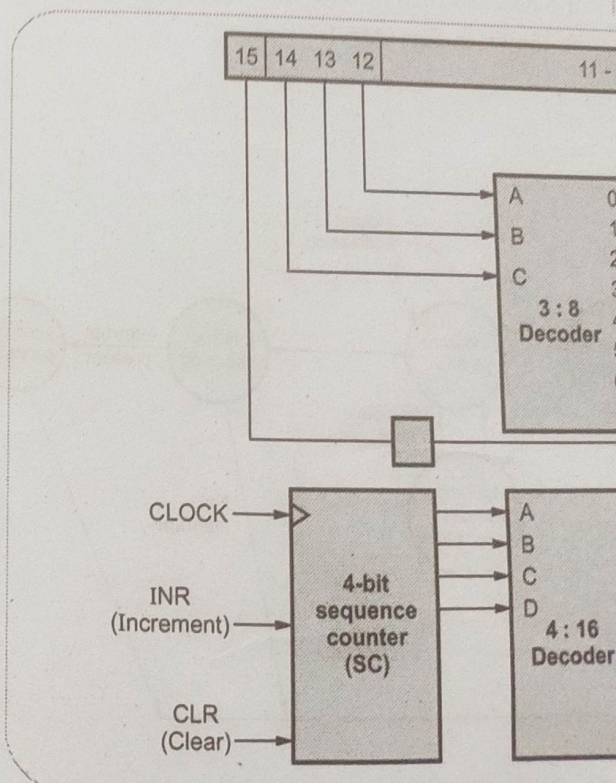


Fig. Q.14.1 (a) Cont

Ans. : The Fig. Q.14.1 (a) shows the block diagram of control unit which uses hardwired approach. It consists of decoders, a sequence counter and combinational logic circuitry.

The Fig. Q.14.1 (a) shows how the instruction in the Instruction Register (IR) is used to generate control signals. The instruction in the instruction register is divided into three fields : I field, operation code field and bits 0 through 11. The 3 : 8 decoder decodes the operation code bits and outputs of this decoder are used as an inputs for combinational logic circuitry. The four bit sequence counter counts from 0000 through 1111, i.e. (0 through 15 in decimal). These four bits are decoded by 4 : 16 decoder to generate timing signals from 0 through 15. The I bit from IR is also connected to the combinational control circuitry. Therefore, combinational control circuitry takes the input from 3 : 18 decoder, 4 : 16 decoder, I bit and bits 0 through 11 to generate control signals.

Along with the clock input Sequence Counter (SC) has two other inputs : Clear and increment. Initially, sequence counter is cleared by activating clear input. Once the SC is cleared, clear signal is negated and SC is incremented with every positive clock transition to produce the sequence of timing signals T_0, T_1, T_2, T_3 and so on. If SC is not cleared, the timing signal will continue with T_4, T_5, up to T_{15} and back to T_0. However, if SC is cleared in between, a new sequence will start from T_0, T_1 and so on, as shown in the Fig. Q.14.1 (b).

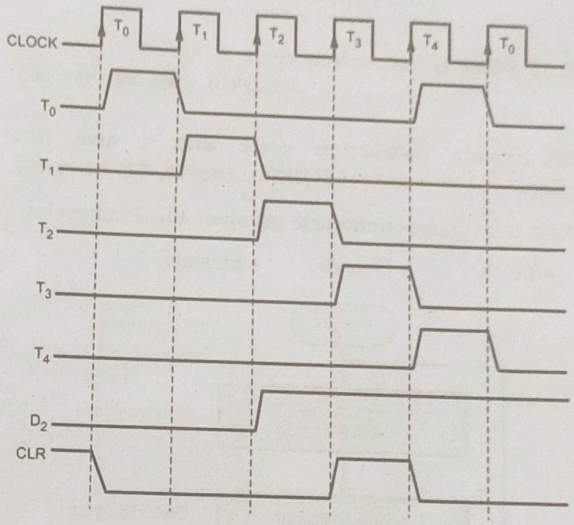


Fig. Q.14.1 (b) Example of control timing signals

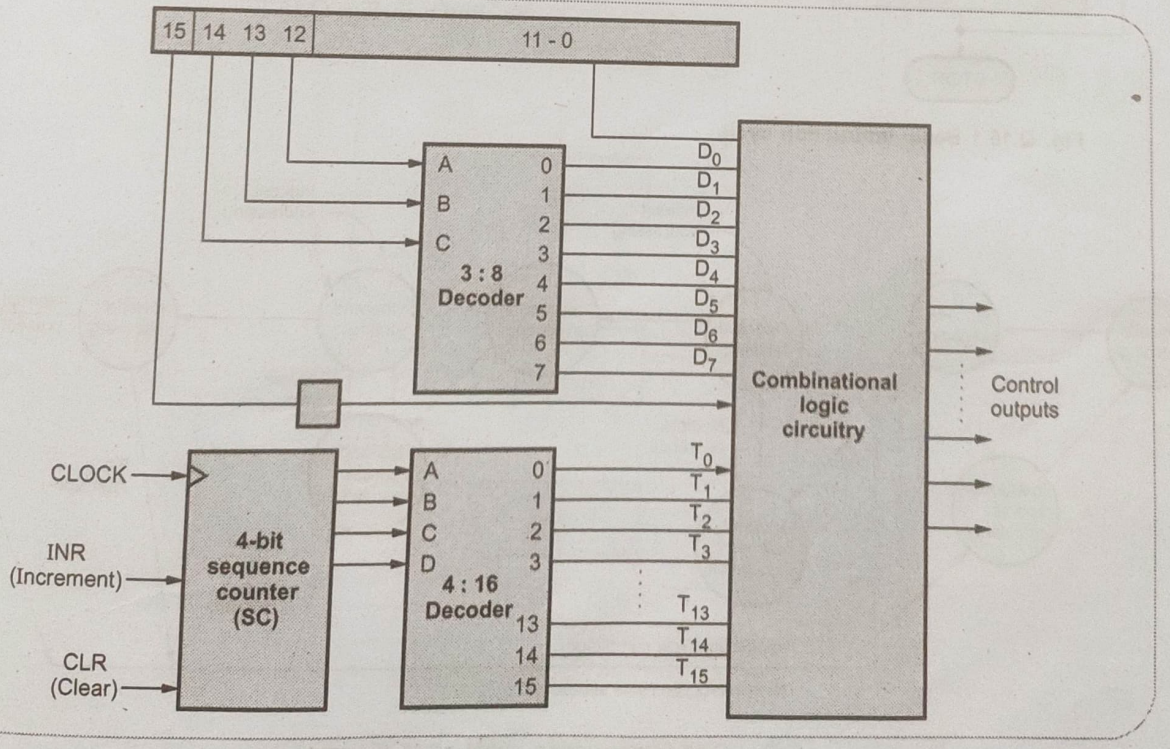


Fig. Q.14.1 (a) Control unit of basic computer

As shown in the Fig. Q.14.1 (b) the SC is cleared to 0 in T_4 if decoder output D_2 is active (logic 1). This can be expressed symbolically as

$$D_2 T_4 : SC \leftarrow 0$$

3.5 : Instruction Cycle

Q.15 What is instruction cycle ?

☞ [JNTU : Dec.-04, Nov.-09]

OR Explain instruction cycle with a neat flow chart.

☞ [JNTU : Dec.-17, 18, Marks 10]

Ans. : An instruction cycle involves three subcycles,

- Fetch
- Decode
- Execute.

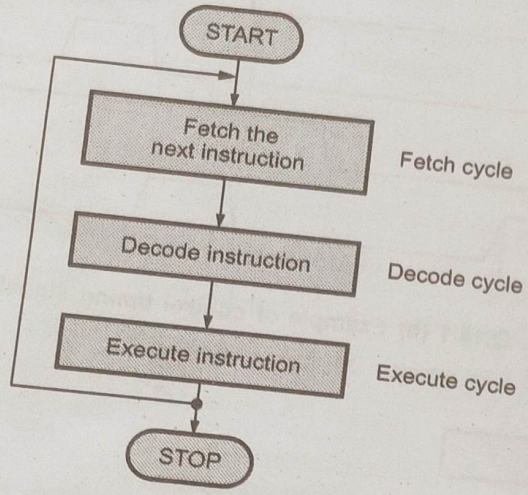


Fig. Q.15.1 Basic instruction cycle

- The fetch phase reads the next instruction from memory into the CPU.
- The decode phase interprets the opcode by decoding it.
- The execute phase performs the indicated operation.

Q.16 Draw and explain instruction cycle state diagram.

Ans. : If the operands on which the instruction works are present within the CPU-registers, a memory access is not required. But if the execution of an instruction involves one or more operands in memory, each requires a memory access. If indirect addressing is used, then additional memory accesses are required. For fetching the indirect addresses, one or more instruction sub cycles are required. After fetching the instruction, it is decoded and if any indirect addressing is involved, the required operands are fetched using indirect addressing. Also, after performing the operation on the operands according to the opcode, a similar process may be needed to store the result in memory. Following execution, interrupt processing may be required before fetching the next instruction. The same process can be viewed as shown in Fig. Q.16.1.

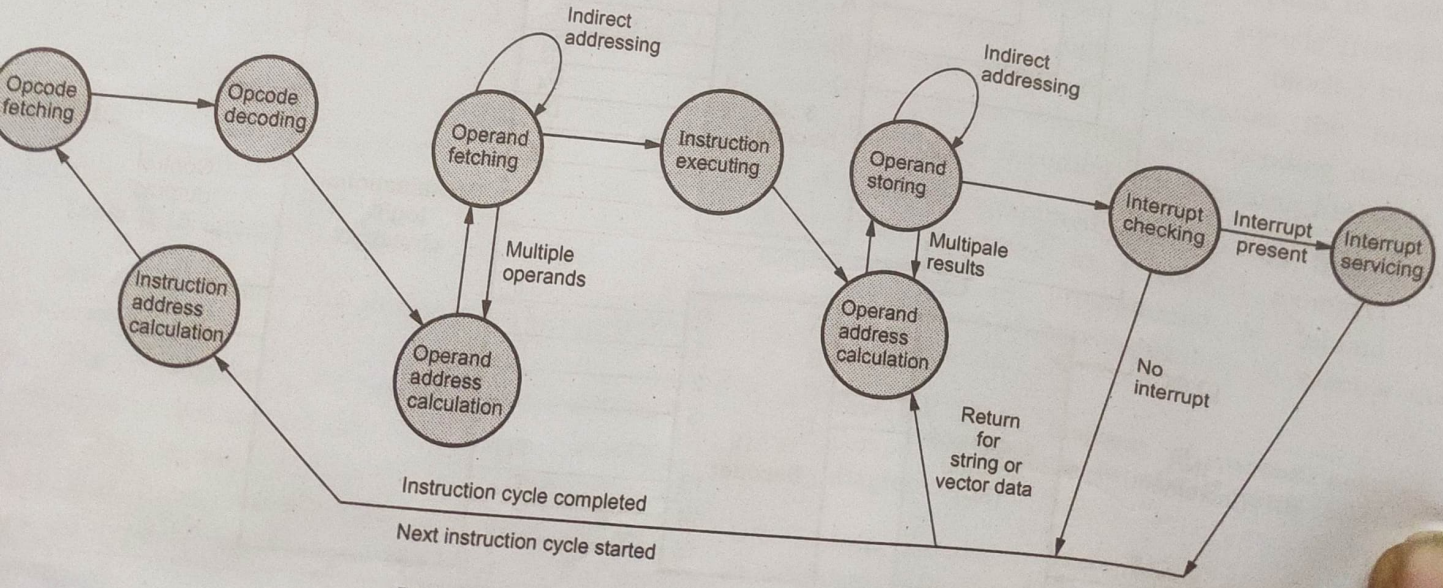


Fig. Q.16.1 Instruction cycle state diagram

Q.29 Draw and explain the flowchart for interrupt cycle.

Ans. : Fig. Q.29.1 shows the flowchart for interrupt cycle in our basic computer organization.

When interrupt flip-flop (R) is 0, computer goes through instruction cycle. After completion of current instruction IEN flag is checked. If it is 0, computer goes to the next instruction cycle. If IEN is 1, control checks flag bit (FGI and FGO). If both flags are 0, it indicates that neither input nor the output registers are ready for transfer of information. In this case computer goes to next instruction cycle. However, if FGI or FGO or both equal to 1, flip-flop R is set to 1. Before proceeding to new instruction cycle, control checks the R flip-flop, if it is 1, it goes to an interrupt cycle instead of an instruction cycle.

In the interrupt cycle, first return address is stored in the memory stack or in a specific memory location. Here we use memory location 0 to store the return address. Here, the vector location address interrupt is 1. Thus PC is loaded with 1 and, IEN and R are cleared to disable further interrupts until current interrupt request is processed.

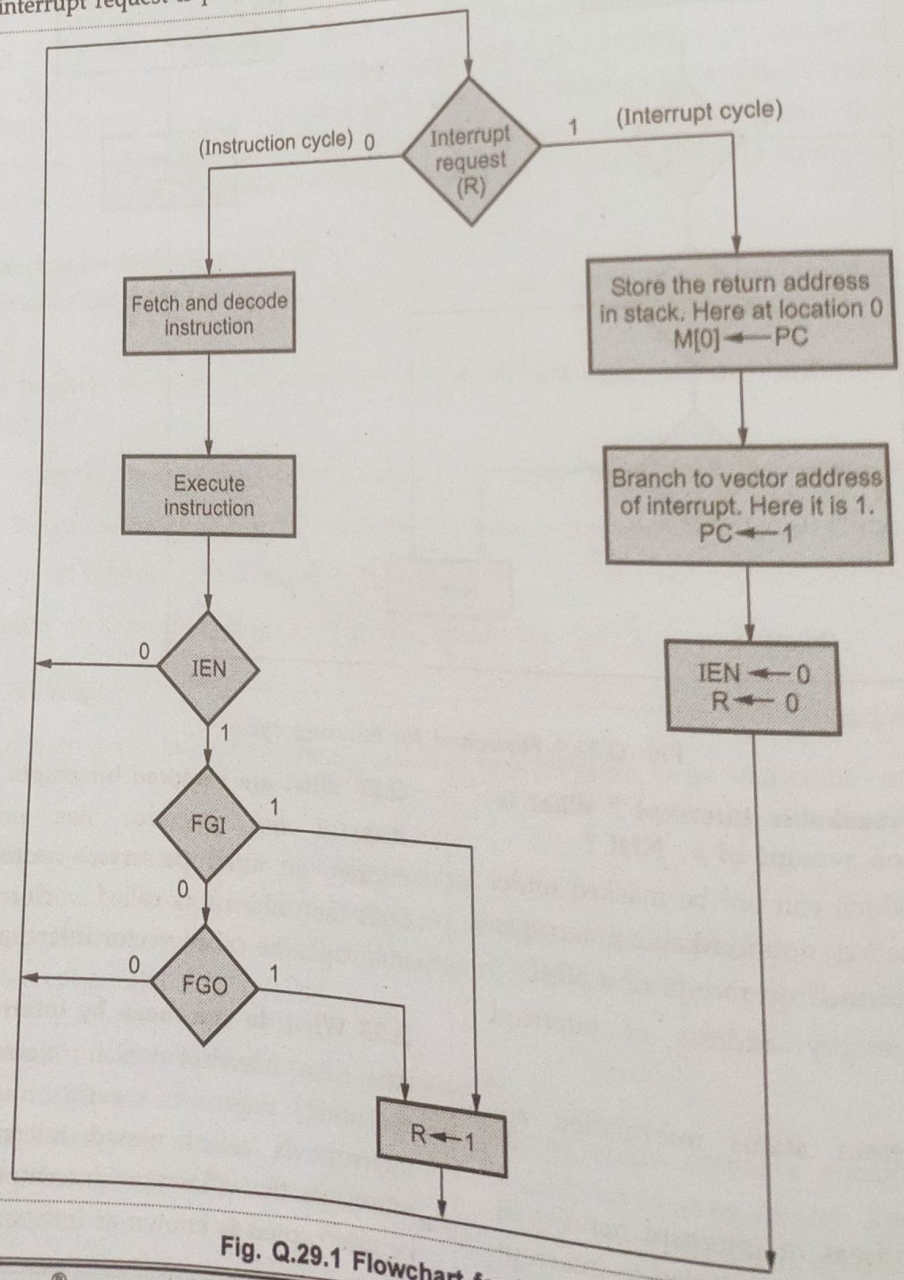


Fig. Q.29.1 Flowchart for int

UNIT - II

4 Microprogrammed Control

4.1 : Control Memory

Q.1 Define the following terms :
i) Microoperation ii) Microinstruction
iii) Microcode iv) Microprogram.

Ans. : i) Micro-operation : To perform fetch, decode and execute cycles, the processor unit has to perform set of operations called micro-operations.

ii) Microinstruction : Each word in the control memory is a microinstruction which specifies the control signals to be activated to perform one or more micro-operations.

iii) Microcode : The translation of symbolic microprogram to binary produces a binary microprogram called microcode.

iv) Microprogram : A sequence of one or more micro-operations designed to perform specific operation, such as addition, multiplication is called a microprogram.

Q.2 Define control memory.

Ans. : Microprogramming is a method of control unit design in which the control signal selection and sequencing information is stored in a ROM or RAM called a **control memory** CM.

Q.3 Define control word. ☞ **[JNTU : Nov.-09, Marks 2]**

Ans. : Grouping the control variables at any given time form a string of 1's and 0's, called a **control word.** The control words are stored in the control memory to perform various operations on the components of the system.

Q.4 Discuss the basic structure of micro program control unit. ☞ **[JNTU : Nov.-03, May-04,19, March-06, Marks 10]**

Ans. : Fig. Q.4.1 shows the microprogrammed control unit. It consists of control memory, control address register micro instruction register and microprogram sequencer.

The components of control unit work together as follows :

- The control address register (μpc) holds the address of the next microinstruction to be read. Every time a new instruction is loaded into the IR, the output of the block labeled "starting address generator" is loaded into the μpc.
- When address is available in control address register, the sequencer issues READ command to the control memory.
- After issue of READ command, the word from the addressed location is read into the microinstruction register.
- The μpc is then automatically incremented by the clock, causing successive microinstructions to be read from the control memory.

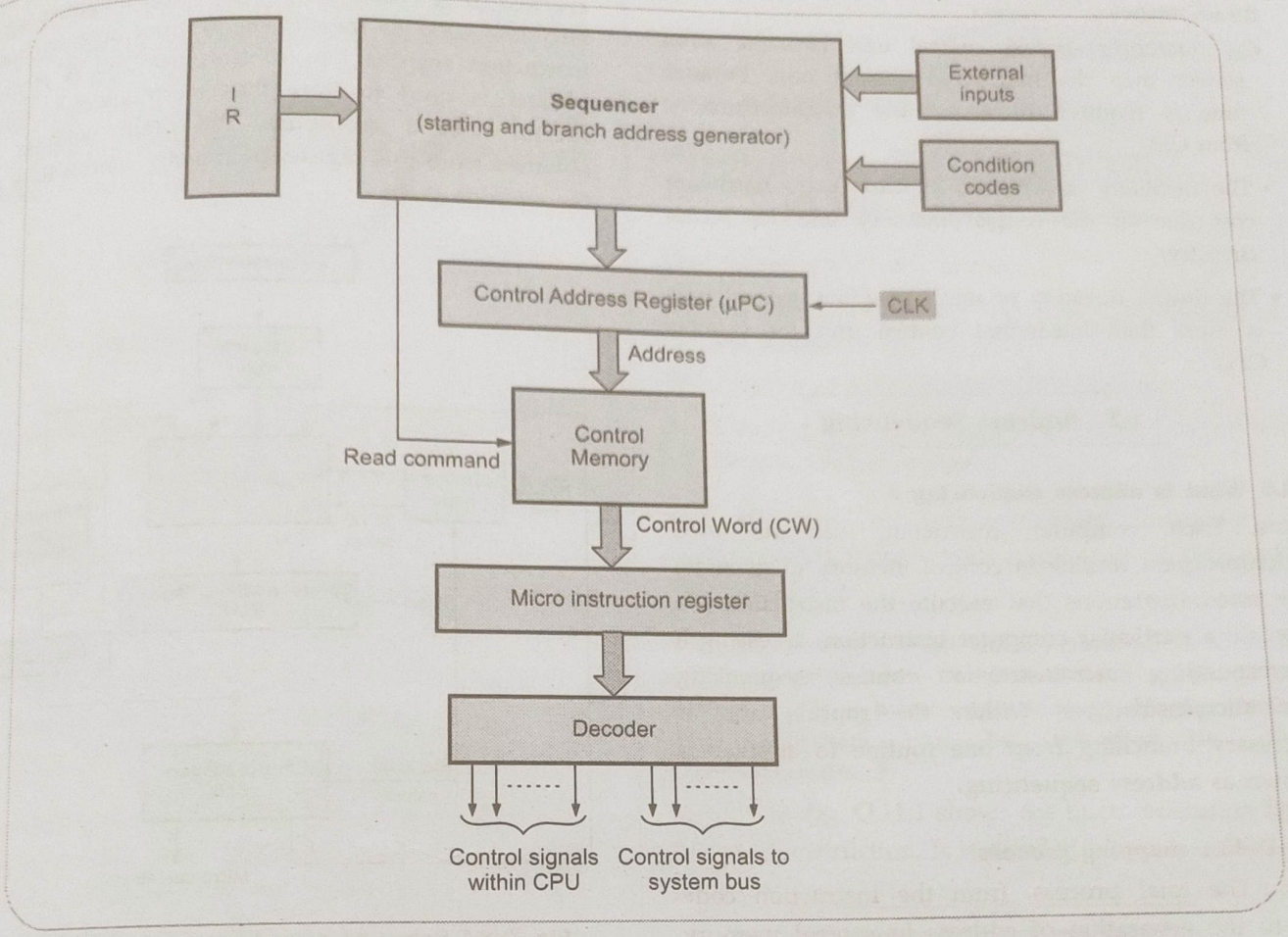


Fig. Q.4.1 Microprogrammed control unit

- The content of the micro instruction register generates control signals which are delivered to various parts of the processor in the correct sequence.
- Number of times the control unit is required to check the status of the condition codes or external inputs to choose between alternative courses of action. In such situation, microprogrammed control use conditional branch microinstructions. In additions to the branch address, these instructions specify which of the external inputs, condition codes, or, possibly bits of the instruction register should be checked as a condition for branching to take place.

Q.5 State the advantages and disadvantages of microprogrammed control unit.

[JNTU : May-19, Marks 4]

Ans. : Advantages :

- It simplifies the design of control unit. Thus it is both, cheaper and less error prone to implement.
- Control functions are implemented in software rather than hardware.
- The design process is orderly and systematic.
- More flexible, can be changed to accommodate new system specifications or to correct the design errors quickly and cheaply.
- Complex function such as floating point arithmetic can be realised efficiently.
- The new or modified instruction set of CPU can be easily implemented by simply rewriting or modifying the contents of control memory.
- The fault can be easily diagnosed in the micro-program control unit using diagnostics tools by maintaining the contents of flags, registers and counters.

Disadvantages :

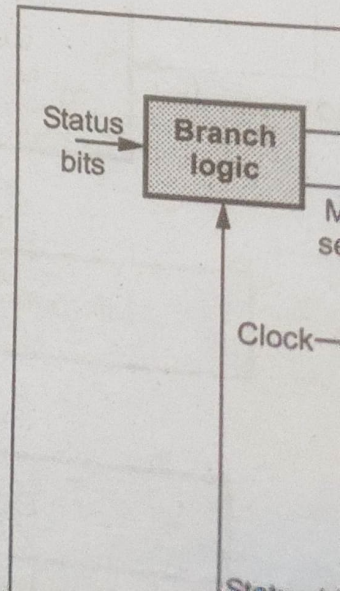
- A microprogrammed control unit is some what slower than the hardwired control unit, because time is required to access the microinstructions from CM.
- The flexibility is achieved at some extra hardware cost due to the control memory and its access circuitry.
- The design duration of micro-program control unit is more than hardwired control unit for smaller CPU.

4.2 : Address Sequencing

Q.6 What is address sequencing ?

Ans. : Each computer instruction has its own microprogram routine in control memory to generate the micro-operations that execute the instruction. To execute a particular computer instruction, accessing a corresponding microinstruction routine, sequencing the microinstructions within the routine and if necessary branching from one routine to another is known as **address sequencing**.

checked to determ
circuit is used to
instruction registe
register is used
execution of a
obtained from thi
the microprogram



checked to determine its condition. A mapping logic circuit is used to transfer an external address from instruction register into control memory. A special register is used to store the return address. After execution of a subroutine, the return address is obtained from this register to transfer control back to the microprogram.

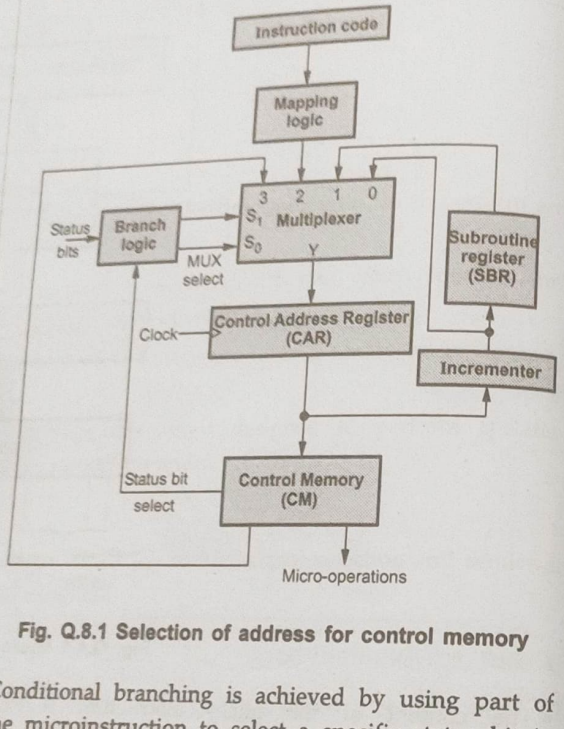


Fig. Q.8.1 Selection of address for control memory

Conditional branching is achieved by using part of the microinstruction to select a specific status bit in order to determine its condition. Special bits are used to check conditions such as the sign bit of a number, carry-out of an adder, the mode bits in an instruction and input or output status conditions. The branch logic checks the status of these bits (1 or 0) together with the field in the microinstruction that specifies a branch address and control the conditional branch decisions.

The branch logic hardware checks the status of bits reserved in the microinstruction to take branching decision on the occurrences of specified conditions.

Q.9 How do you map micro-operation to a microinstruction address ?

[JNTU : Feb.-07, 08, Marks 8]

Ans. : Each instruction of a computer has its operation code (opcode). For each opcode, there is a microprogram routine in control memory that is to be executed for the execution of an instruction. Fig. Q.9.1 shows the mapping process that converts the 4-bit opcode to a 7-bit address for accessing microprogram routine in a control memory.

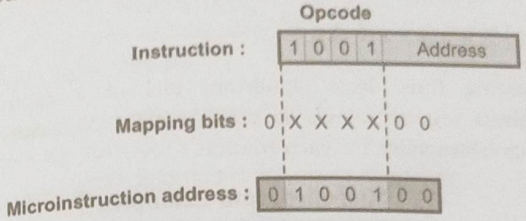


Fig. Q.9.1 Mapping of instruction

This mapping process includes,

1. Placement of 0 in the most significant bit of the address.
2. Transfer of 4-bit operation code in the address.
3. Clearing the two least significant bits of the control address register.

This mapping provides a microprogram routine with a capacity of four microinstructions for each instruction. However, if the microprogram routine needs more than four microinstructions, it can use addresses 1000000 through 1111111 or unused memory locations of instructions which uses fewer than four microinstructions.

Q.10 Formulate a mapping procedure that provides eight consecutive microinstructions for each routine. The operation code has six bits and the control memory has 2048 words.

Ans. : Each instruction of a computer has its operation code (opcode). For each opcode, there is a microprogram routine in control memory that is to be executed for the execution of an instruction.

It is given that the opcode has 6-bits. Since the control memory has 2048 (2^{11}) words, the address of control memory is 11-bits.

The Fig. Q.10.1 shows the mapping process that converts the 6-bit opcode to a 11-bit address for accessing microprogram routine in a control memory.

Q.15 Formulate a mapping procedure that provides eight consecutive micro instructions for each routine. The operation code has 7 bits and control memory has 4096 words.

☞ [JNTU : April-18, Marks 5]

Ans. : Opcode (operation code)=7 bits

Control memory address = $\log_2 4096 = 12$ bits

0 0 x x x x x x x 0 0 0

Clearing three least significant bits of a control address register allows to provide eight consecutive microinstructions for each routine.

4.3 : Microprogram Examples

Q.16 Draw and explain the microinstruction format.

Ans. : Fig. Q.16.1 shows the microinstruction format for the control memory.

As shown in Fig. Q.16.1, the microinstruction includes four fields.

3	3	3	2	2	7
F1	F2	F3	CD	BR	Address (AD)

Fig. Q.16.1 Microinstruction format

1. **F1, F2, F3 :** These are micro-operation fields. Each field is of three-bits. They specify micro-operations for the computer.
2. **CD :** This two-bit field selects status bit conditions for branching operation. The condition includes zero value in AC, sign bit of AC equal to 1 or 0, etc.
3. **BR :** This 2-bit field specifies the type of branch to be used. Branch type includes unconditional branch, branch if zero, branch if negative and so on.
4. **AD :** This is an address field which contains a branch address. This field is of seven bits since control memory has 128 words. ($128 = 2^7$).

4.4 : Design of Control Unit

Q.17 With the help of neat diagram explain the decoding of micro-operation fields and their further processing.

Ans. : Fig. Q.17.1 shows the block diagram to decode micro-operation fields and their further processing. (see Fig. Q.17.1 on next page.)

The functions of blocks in Fig. Q.17.1 are explained below.

3 × 8 decoders

The control memory output of each subfield in microinstruction are decoded with a 3 × 8 decoder to provide eight outputs.

These outputs are connected to the proper circuit to initiate the corresponding micro-operation.

Arithmetic logic shift unit

The arithmetic logic shift unit is designed to perform arithmetic, logical and shift operations. The output of the decoders are used to select the appropriate operation in the arithmetic logic shift unit. The Fig. Q.17.1 shows some of the outputs of decoders connected to the arithmetic logic shift unit. The other outputs of the decoders that are associated with an AC operation are also connected (not shown in the Fig. Q.17.1) to the arithmetic logic shift unit.

Multiplexer

Multiplexer is used to select the source for the AR register. When select input is 0, multiplexer selects the contents of PC as a source, otherwise it uses the contents of DR as a source.

Q.18 Explain microprogram sequencer with block diagram.

OR Define microprogram sequencer.

☞ [JNTU : Dec.-19, Marks 5]

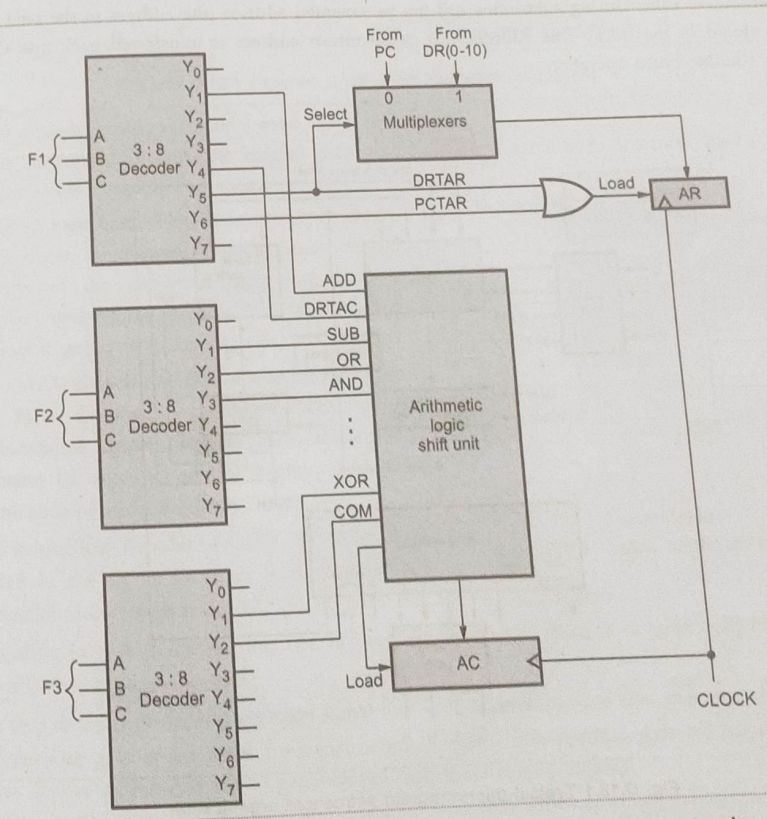


Fig. Q.17.1 Decoding of micro-operation fields and their further processing

Ans. : • The subunit of the microprogrammed control unit which presents an address to the control mem called **microprogram sequencer.** The next-address logic of the sequencer determines the specific address so be loaded into the control address register.

• The Fig. Q.18.1 shows the block diagram of commercial microprogram sequencer. It consists o multiplexer that selects an address from four sources and routes it into a control address register. T output from CAR provides the address for the control memory. The contents of CAR are incremen and applied to the multiplexer and to the stack register file. The register selected in the stack determined by the stack pointer. Inputs I_2, I_1, I_0 and T derived from the CD and BR fields microinstruction specify the operation for the sequencer. They specify the input source to multiplexer also generate push and pop signals required for stack operation. The stack pointer three-bit register and it gives the address of stack register file consists of $(2^3 = 8)$ eight regist Initially, the stack pointer is cleared and is said to point at address 0 in the stack. Using p operation it is possible to write data into the stack at the address specified by the stack pointer. A data is written, stack pointer is incremented by one to get ready for the next push operations.

• In pop operation stack pointer is decremented by one and then the contents of the register specifi by the new value of stack pointer are read. With this mechanism it is possible to implem subroutine calls. During subroutine call the incremented address (the address of the next instructio is stored in the stack. This address also called **return address** is transferred back into CAR with subroutine return operation.

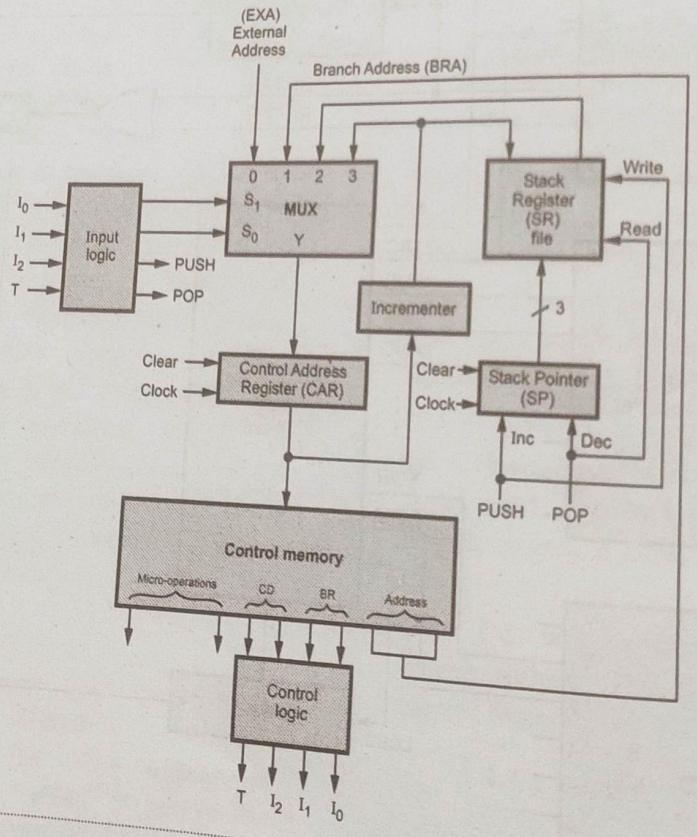


Fig. Q.18.1 Typical microprogram sequencer organization

• The Table Q.18.1 address is trans $S_1S_0 = 01$ and T $S_1S_0 = 11$, increm

• A call to subrou operation and a executed by act the address stor

I_2	I_1	I_0
X	0	0
1	0	1
0	0	1
X	1	0
0	1	1

Q.19 Draw and explai

Ans. : • In the hardw generate cont

• The fixed log step counter, contents of external inpu requests to g

• Fig. Q.19.1 unit. Here, combination generates th on the state

• The instruc loaded in instruction

• According and all othe

• The step de

• The encode codes. It us

• After execu it ready fo

- The Table Q.18.1 gives the function table for microprogram sequencer. When S_1S_0 = 00, an external address is transferred to CAR. The transfer from address field of microinstruction occurs when S_1S_0 = 01 and T = 1. When S_1 S_0 = 10, stack register contents are transferred to CAR and when S_1S_0 = 11, incremented contents of CAR are transferred to the CAR
- A call to subroutine is executed by activating push signal when S_1S_0 = 01. This causes a push-stack operation and a branch to the address specified by microinstruction. The return from subroutine is executed by activating pop signal when S_1S_0 = 10. This causes a pop-stack operation and a branch to the address stored on top of the stack.

I_2	I_1	I_0	T	S_1	S_0	Operation	Description
X	0	0	X	0	0	CAR ← EXA	Transfer external address.
1	0	1	1	0	1	CAR ← BRA , SR ← CAR + 1	Branch to subroutine and save the next instruction address in stack (Push operation).
0	0	1	1	0	1	CAR ← BRA	Transfer branch address.
X	1	0	X	1	0	CAR ← SR	Transfer from stack register.
0	1	1	0	1	1	CAR ← CAR + 1	Increment address.

Table Q.18.1 Function table of microprogram sequencer

Q.19 Draw and explain hardwired control unit.

Ans. :

- In the hardwired control, the control units use fixed logic circuits to interpret instructions and generate control signals from them.
- The fixed logic circuits use contents of the control step counter, contents of the instruction register, contents of the condition code flag and the external input signals such as MFC and interrupt requests to generate control signals.
- Fig. Q.19.1 shows the typical hardwired control unit. Here, the fixed logic circuit block includes combinational circuit (decoder and encoder) that generates the required control outputs, depending on the state of all its inputs.

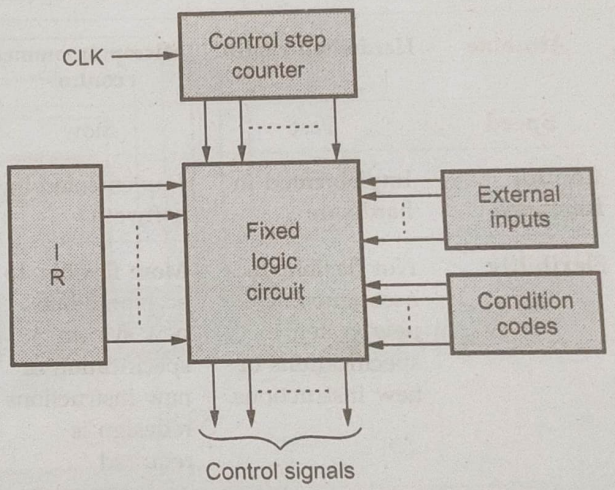


Fig. Q.19.1 Typical hardwired control unit

- The instruction decoder decodes the instruction loaded in the IR. If IR is an 8-bit register then instruction decoder generates 2^8, i.e. 256 lines; one for each instruction.
- According to code in the IR, only one line amongst all output lines of decoder goes high i.e., set to 1 and all other lines are set to 0.
- The step decoder provides a separate signal line for each step or time slot, in a control sequence.
- The encoder gets in the input from instruction decoder, step decoder, external inputs and condition codes. It uses all these inputs to generate the individual control signals.
- After execution of each instruction end signal is generated which resets control step counter and make it ready for generation of control step for next instruction.

Q.20 State the advantages and disadvantages of hardwired control unit.

Ans. : **Advantages :**

- Hardwired control unit is fast because control signals are generated by combinational circuits.
- The delay in generation of control signals depends upon the number of gates.
- It has greater chip area efficiency since its uses less area on-chip.

Disadvantages :

- More the control signals required by CPU; more complex will be the design of control unit.
- Modifications in control signal are very difficult. That means it requires rearranging of wires in the hardware circuit.
- It is difficult to correct mistake in original design or adding new feature in existing design of control unit.

Q.21 Compare hardwired control unit and microprogrammed control unit.

☞ [JNTU : Dec.-19, Marks 5]

Ans. :

Attribute	Hardwired control	Microprogrammed control
Speed	Fast	Slow
Control functions	Implemented in hardware	Implemented in software
Flexibility	Not flexible, to accommodate new system specifications or new instructions.	More flexible, to accommodate new system specification or new instructions redesign is required.
Ability to handle large/complex instruction sets	Somewhat difficult	Easier
Ability to support operating systems and diagnostic features	Very difficult (unless anticipated during design)	Easy
Design process	Somewhat complicated	Orderly and systematic
Applications	Mostly RISC microprocessors	Mainframes, some microprocessors
Instruction set size	Usually under 100 instructions	Usually over 100 instructions
ROM size	–	2 K to 10 K by 20-400 bit microinstructions
Chip area efficiency	Uses least area	Uses more area

Q.22 It is possible to have a hardwired contro associated with a control memory.

☞ [JNTU : Dec.19, Mark 1]

Ans. : No, it is not possible to have a hardwire control associated with a control memory because, b definition, it does not contain a control memory.

Q.23 It is possible to design a microprocesso without a microprogram ? Are all microprogramme computers also microprocessors ?

☞ [JNTU : Dec.-19, Marks 3]

Ans. : Microprogram is a program for a sequence o microoperations. The control unit of a microprocesso can be hardwired or microprogrammed, dependin on the specific design. A microprogrammed compute does not have to be a microprocessor.

END...