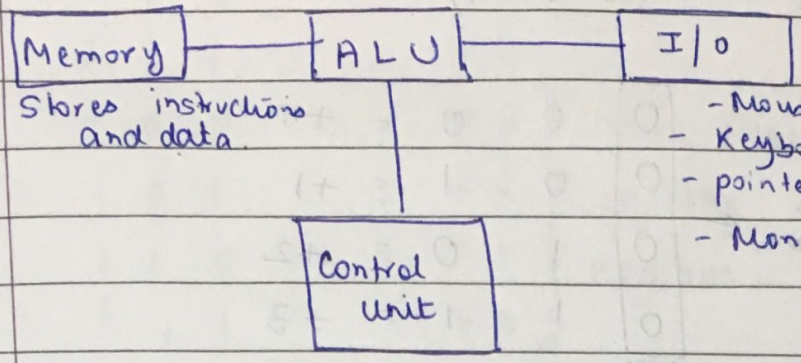
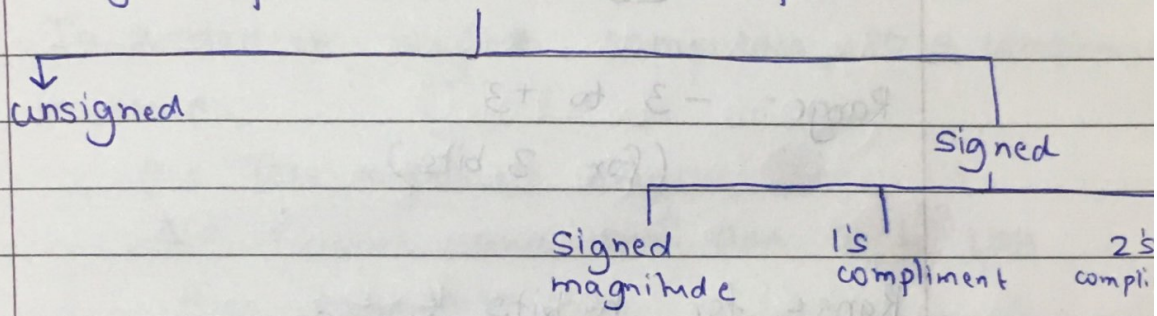
Date

Computer Organization and Architecture (COA)



- facilitates working of other functional unit
- generate control signals

* Integer representation in a computer



Unsigned integers contain only magnitude information (scalar value)

$5_{(10)} = (101)_2$

radix format

The decimal no. range is 0 to $2^n - 1$.

Signed integers: It contains both the magnitude as well as the sign of the integer.

+5 — MSB (Most significant bit)

0 = +ve

1 = -ve

Example : +5 0101

-5 1101

3 bits (0 to 7)

positive numbers ←

0	0	0	= +0
0	0	1	= +1
0	1	0	= +2
0	1	1	= +3

negative numbers ←

1	0	0	= -0
1	0	1	= -1
1	1	0	= -2
1	1	1	= -3

Range : -3 to +3
(for 3 bits)

Range for n bits :

$-(2^{n-1}-1)$ to $(2^{n-1}-1)$

* 1's compliment

We look at the MSB first, if MSB is 0 then the no. is +ve else the no is -ve.
We only flip the negative numbers and not the positive ones.

Page No.
Date

0 0 0	=	+0
0 0 1	=	+1
0 1 0	=	+2
0 1 1	=	+3
1 0 0	=	-3
1 0 1	=	-2
1 1 0	=	-1
1 1 1	=	-0

We take the magnitude as the inital no. is negative.

The maximum decimal range for n bits is
$-(2^{n-1}-1)$ to $(2^{n-1}-1)$

* In a modern days computers, 2's complimen is used.

0 = Take magnitude only
1 = Flipped and add 1 to LSB then take magnitude.

0	0	0	=	0
0	0	1	=	1
0	1	0	=	2
0	1	1	=	3
1	0	0	=	-4
1	0	1	=	-3
1	1	0	=	-2
1	1	1	=	-1

Range : $-(2^{n-1})$ to $(2^{n-1}-1)$

	unsigned	signed	1's comp	2's comp
+5	0101	0101	0101	0101
-5	0101	1101	1010	1011

BCD - Binary Coded Decimal

They are represented only in 4 bit binary number. It is a weighted code.

It is also known as 8421 code.

(0000 to 1001)

(0 to 9).

Q $(35)_{10}$ into BCD

$(\underbrace{0011}_{3}\ \underbrace{0101}_{5})$

- Binary is more efficient than BCD.

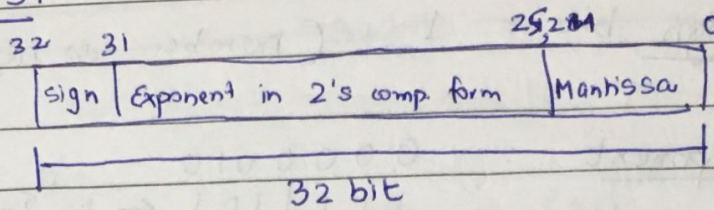
Example: $(2479)_{10}$ into BCD

0010 0100 0111 1001

Page No.
Date

* Floating point representation
  - ANSI format
  - IEEE 754 format

ANSI



0.6132×10^{5} → exponent

Mantissa — base

Mantisa - 24 bits
exponent - 8 bits
sign - 1 bits

This can be represented as.

$$M \times b^{e}$$

Example: Store - 0.125 in ANSI floating point representation.

Step 1: Convert - 0.125 into binary values.

$0.125 \times 2 = 0.25$
$0.25 \times 2 = 0.5$
$0.5 \times 2 = 1$

Step 2: Normalisation.

A floating point is said to be normalised if the most significant bit of mantissa is non-zero.

Binary equivalent of - 0.125 is $-(0.001)_2$

After normalisation,

-0.1×2^{-2}

$-0.(1) \times 2^{(-10)}$ → Binary equivalent of 2.

$M \times 8^{e}$

Sign bit = 1 (number is negative)

Exponent : 0 0 0 0 0 1 0

1 1 1 1 1 0 1 ← if it would be a true no then we would have taken this.

+1

1 1 1 1 1 1 0

as it is a negative number then we will take this

In exponent we will only consider the exponential power.
i.e. in this case we will take -2.

Mantissa

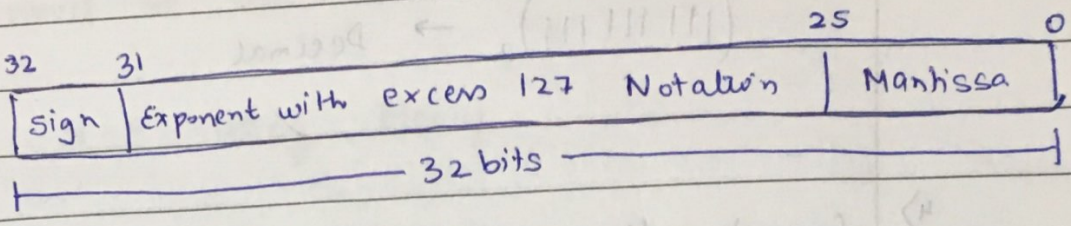
1 followed by 23 zeros.

ANSI representation : 1 1 1 1 1 1 0 | 0 0 0 0 0 0 0 0 0

23 zeros.

Page No.
Date

* IEEE 754 format



Ques: 1) Convert Decimal to Binary $(23)_{10}$
2) Convert Binary to Decimal $(1001)_2$
3) Convert Octal to Decimal $(777)_8$
4) Convert Hexadecimal to Decimal $(E1)_{16}$

Ans: 1)

2	23	
2	11	1
2	5	1
2	4	1
2	2	0
	1	

$(23)_{10} = 10111$

2) 1001
$3\ 2\ 1\ 0$

$2^3 + 2^0$

$8 + 1$

9

$2^3 \times 2^2 \times 0 + 2^1 \times 0 + 2^0$

$= 9$

Date

We can write 7 as 111 (binary)

3) $(777)_8$ → Octal

$(111\ 111\ 111)_2$ → Decimal

4) Convert Hexadecimal into Decimal $(E1)_{16}$.

$(1110\ 0001)_2$

This can be done by:

value of E = 14

$14 \times 16 + 1 \times 16^0$

$= 225$

Q Perform 6-4 using 2's compliment form.

6+(-4)

6 → 0110

0 1 1 0 — 6

0 1 0 0 — 4

2's complement of (-4): -4 = 1100

1011

1

1100

~~0100~~

~~011~~ ~~(1's complement)~~

~~+ 1~~

~~1100~~

Then we will add this

6 + (-4) = 0110

1100

(1) 0010

carry is discarded

If a carry is generated that means the result is +ve and it is discarded.

Ques: 4-6 using Binary.

4 → 0100

6 → 0110

4 + (-6)

-6 → 1000

6 ⟹ 0110

1001 (1's compliment)

1

1010

0100

1010

1110

0001 (1st compliment)

+ 1

10

If no carry is generated then the result is -ve then the 2's compliment of result will give the correct answer.

Q Convert $(ABC)_{16}$ to Octal.

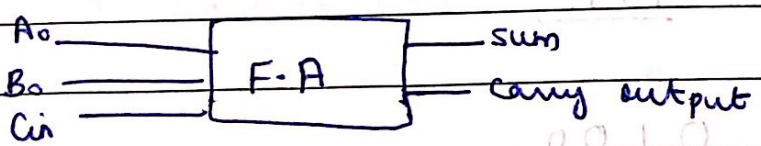
A → 10 1010
B → 11 1011
C → 12 1100

We will make a group of 3 from right

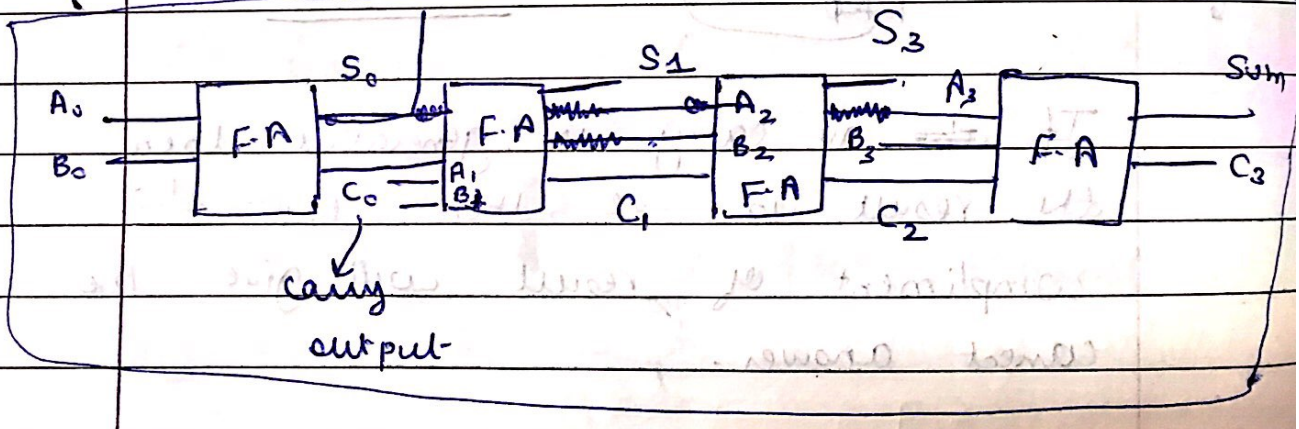
1010101111100

5 2 7 4 ←

Ques: Design a 4 bit binary Adder using full Adder.



We will be using 4 full adder Binary adder.



In full adder, two bits are added along with carry at one time whereas binary adder

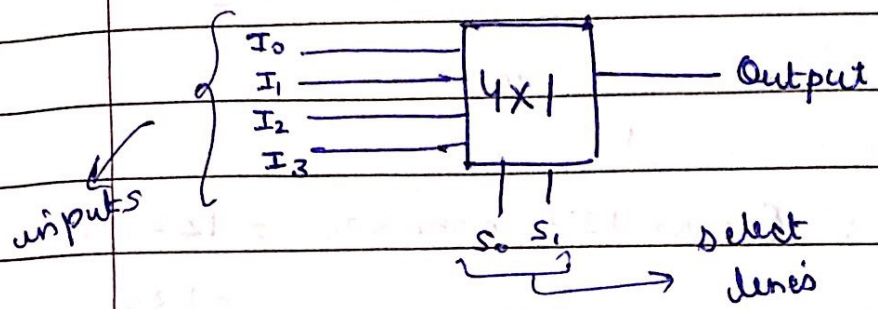
* $A + B' + 1$

$A + (B' + 1)$

$A + (-B)$

$A - B$

Ques: Truth Table of Multiplexer.



S_0	S_1	Output
0	0	I_0 ~~$(S_0 + S_1)$~~
0	1	I_1 ~~$(S_0 + S_1)$~~
1	0	I_2 ~~$(S_0 + S_1)$~~
1	1	I_3 ~~$(S_0 + S_1)$~~

* IEEE 754 standard for floating point representation

Eg: 147.625

① Convert into Binary format = 10010011.101

= 1.0010011101 × 2^7 (7) exponent

mantissa

Exponent : Excess 127 notation = 127 + 7

= 134

= (10000110)$_2$

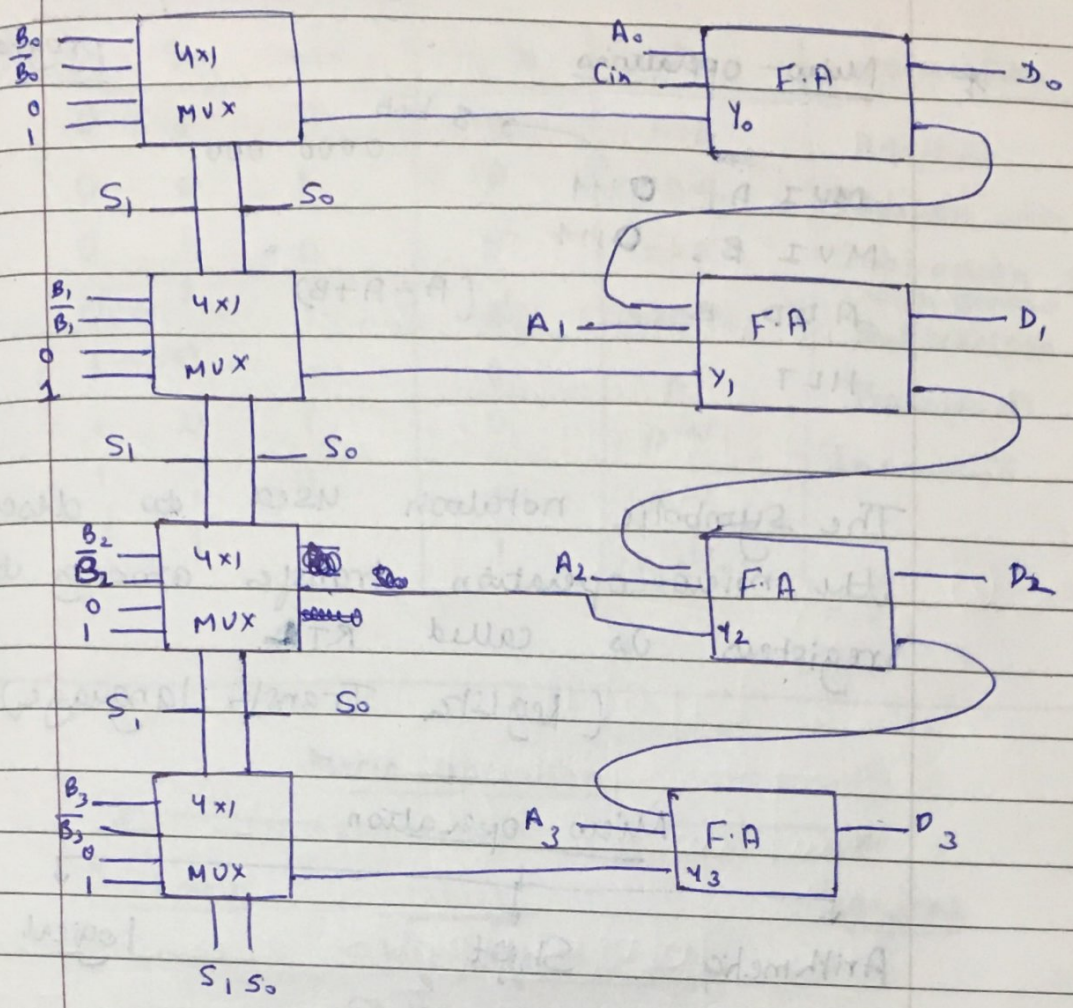
IEEE 754 : 0 10000110 00100111010000...

13

Hardware for arithmetic operations

also known as 4 bit Arithmetic circuit

Page No.
Date



S_1	S_0	Cin	Y	D
0	0	0	B	A+B
0	0	1	B	A+B+1
0	1	0	B'	A+B'
0	1	1	B'	A+B'+1 = A-B
1	0	0	0	A
1	0	1	0	A+1
1	1	0	1	A-1
1	1	1	1	(A+1)+1

↳ This 1 is added to all the bits.

Last case :

1111

↓ 1's (-1)

0000

1

0001 (-1)

A

1111

8 bit
0000 0001

```
MVI A, 0111
MVI B, 0111
ADD B          (A = A+B)
HLT
```

The symbolic notation used to describe the micro-operation transfer among the registers is called RTL.
(Register Transfer language)

Micro operation

Arithmetic	Shift	Logical
A+B	Shift left	
A-B	Right	
A'	Rotate left	
A+1	Rotate right	
A-1	Arithmetic Shift Right	

Page No.
Date

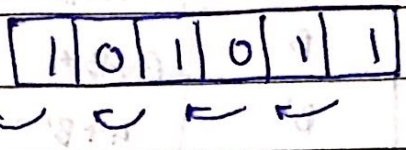
Truth Table of the 4 bit Arithmetic Circuit

S_1	S_0	C_{in}	Y	D	operation
0	0	0	B	A+B	Addition
0	0	1	B	A+B+1	Addition with carry
0	1	0	B'	A+B'	Subtraction on with Borrow
0	1	1	B'	A+B'+1	Subtraction
1	0	0	0	A	Transfer A
1	0	1	0	A+1	Increment
1	1	0	1	A-1	
1	1	1	1	(A+1)+1	

Micro operation

- Arithmetic
- Shift
  - Shift left
  - Shift Right
  - Circular Shift left
  - Circular shift right
  - Arithmetic Shift left
  - Arithmetic Shift right
- logical

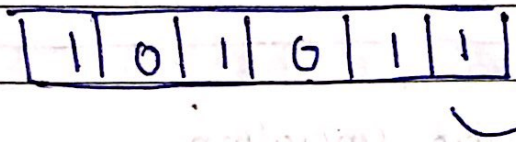
① Shift left micro-operation



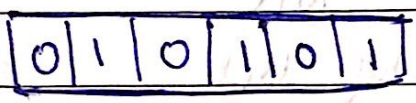
One bit will be lost

padded zero

② Shift Right

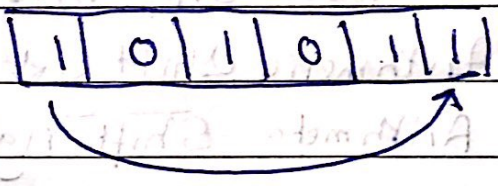


Here also one bit will be lost



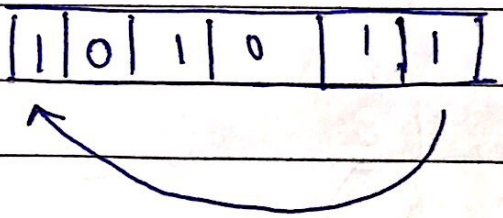
padded zero

③ Circular Shift left



010111

④ Circular Shift right



110101

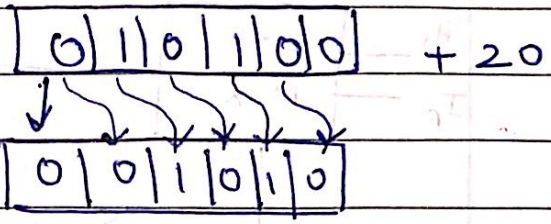
Page No.
Date

Arithmetic

Arithmetic shift is a micro-operation that shifts a signed binary no. to the left or right. (It is a divide by 2 micro-operation.)

① Arithmetic Shift ~~left~~ Right.

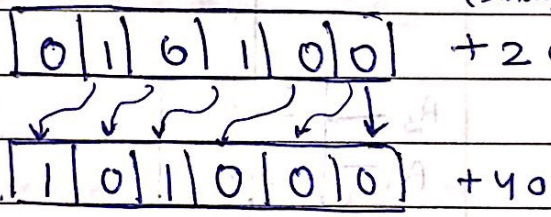
(ashr R)



~~In arithmetic~~

② Arithmetic Shift left

(ashl R)



It is multiply by 2 micro-operation.

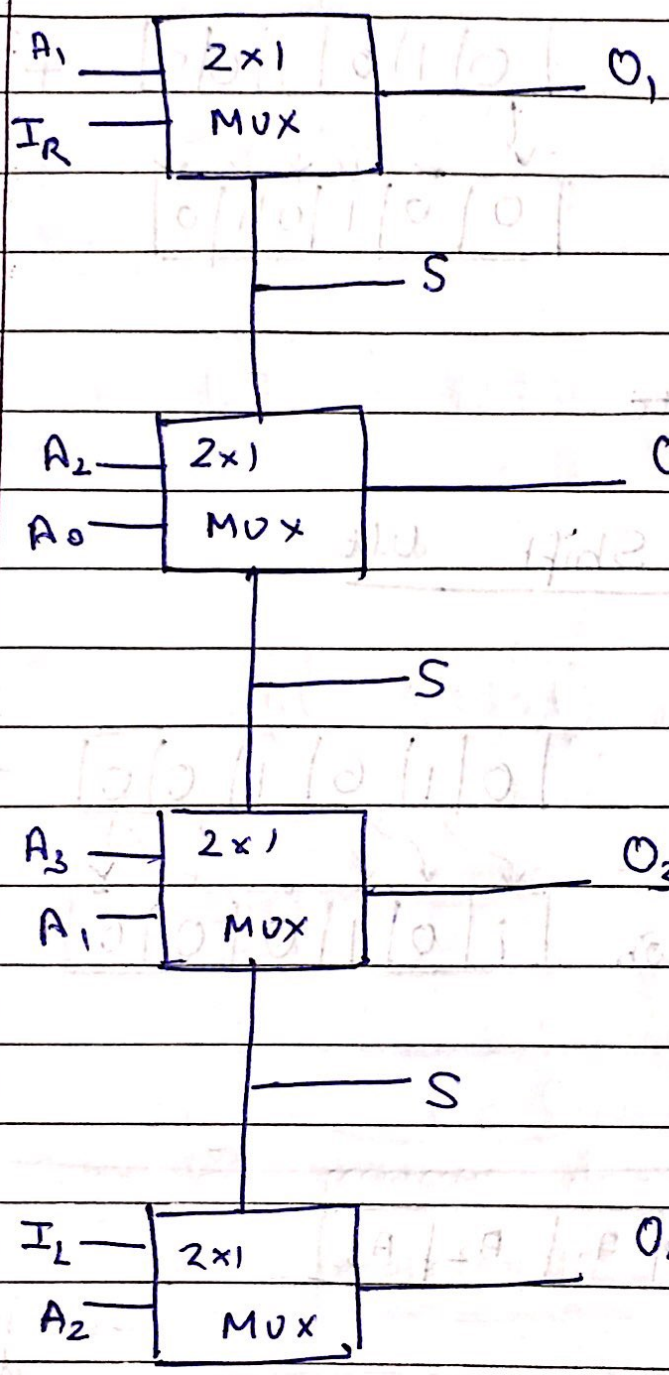
Ques: R A_0 A_1 A_2 A_3

I_L = inserted ~~left~~ bit

(Shift R) Shl R A_1 A_2 A_3 I_L

Shr R I_R A_0 A_1 A_2

I_R: inserted right bit



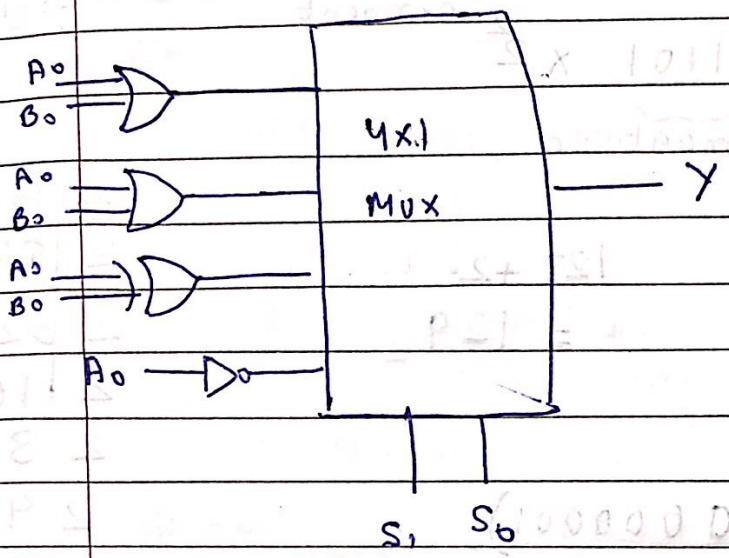
Truth Table

S	Output Operation
0	ShL R
1	Shr R

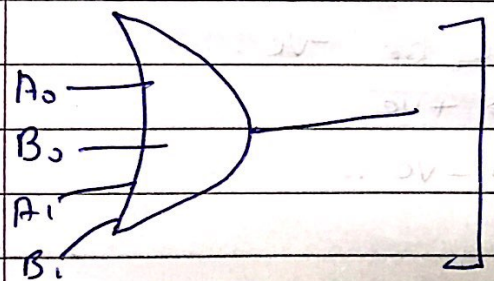
Page No.
Date

* Logical Micro-operation

- OR
- AND
- XOR
- NOT



The above diagram was for one bit. If we want to apply operation on 4 bit number then we uses 4 4x1 MUX multiplexers.



we cannot use this.

Convert 5.625 into IEEE 754 floating point representation.

Step 1 5.625 : Convert this into binary format.

$0.625 \times 2 = 1.250$

$0.250 \times 2 = 0.500$

$0.500 \times 2 = 1.000$

101.101

Step 2: 1.01101×2^{2} (2) exponent

mantissa

Excess 127 notation: 127 + 2 = 129

2	129
2	64 1
2	32 0
2	16 0
2	8 0
2	4 0
2	2 0
	1 0

Step 3: $(10000001)_2$

IEEE 754 : 0 10000001 01101 000...0 (18 zeros)

↑ sign bit

Sign bit denotes whether the no. is positive or -ve.

0 → +ve

1 → -ve.

Page No.
Date

15

Ques: $(438F\ 0000)_{16}$ —IEEE 754→ Decimal equivalent

each no. should have 4 bits

Step 1: Convert it into binary.

2 | 15
2 | 7 — 1
2 | 3 — 1
2 | 1 — 1
0

0100 0111 1000 1111 0000 0000 0000 0000

4 3 8 F 4 0 0 0 0

8 23

4 + 2 + 1.

sign bit

1 1 2 2

3 + 4 + 2 + 1.

In IEEE format,

1 | 8 | 23

Sign | exponent | Mantissa

$1.\text{Mantissa} \times 2^{\text{exponent}}$

Sign = 0

Exponent = 10000111

Excess 127 = 135 (decimal equivalent of exponent).

exponent: 135 − 127

= 8

$1.00011110\ldots0 \times 2^{8}$

16 zeros

Shift 8 zeros

1000 1111 0.000

decimal equivalent

256 + 16 + 8 + 4 + 2

(286.000)

* R = | A_0 | A_1 | A_2 | A_3 |

ashr R | A_0 | A_0 | A_1 | A_2 |

R = 110100 (-12) →

ashr R = 111010 (-6) →

It is divide by 2 operation

$2^5 + 2^4 + 2^2$

$32 + 16 + 4$

$32 + 20$

52

$2^5\ 2^4\ 2^3\ 2^2\ 2^1$

$32 + 16 + 8 + 2$

$32 + 18 + 0$

26

110100

↓ -ve

2's complement nikalo.

01011

1

01100 → 12

111010

↓ -ve

00101

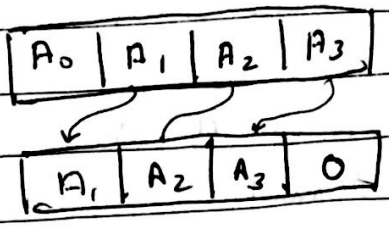
1

110

$2^3 + 2$

Page No.
Date

ashl R



Sign information remains same

R = 110100 -12

ashl R = 1 101000 -24

Sign

110100

10100
01011
1
01100

$2^3\ 2^2\ 2^1\ 2^0$

8×4 32

101000
010111
1
011000

$2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$2\times2\times2\times2\times2$
16×8
128

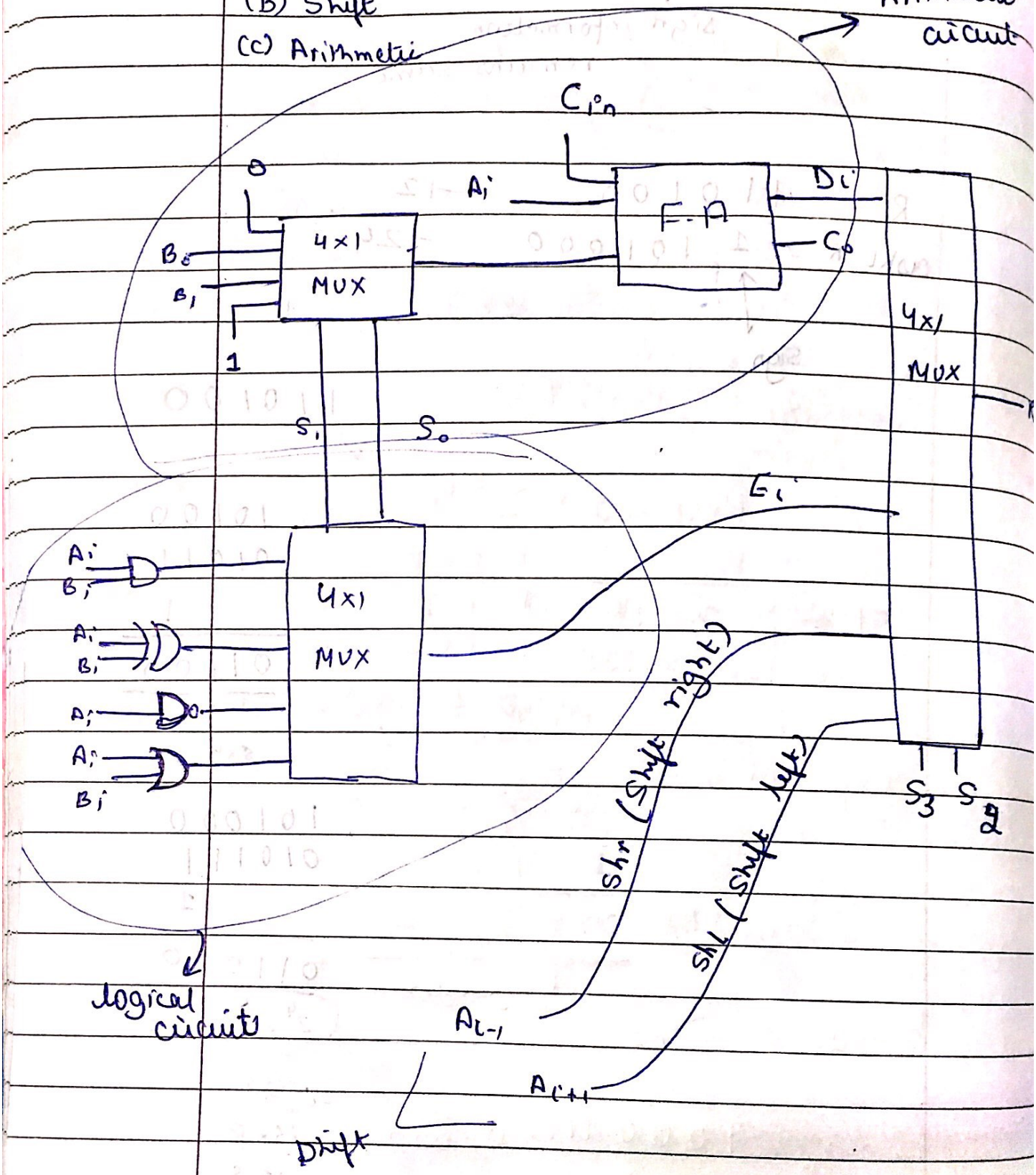
Arithmetic Shift logic Unit (ALU)

Combination of three micro-operations

(A) logical

(B) Shift

(C) Arithmetic



One stage of Arithmetic logic Shift Unit (ALU).

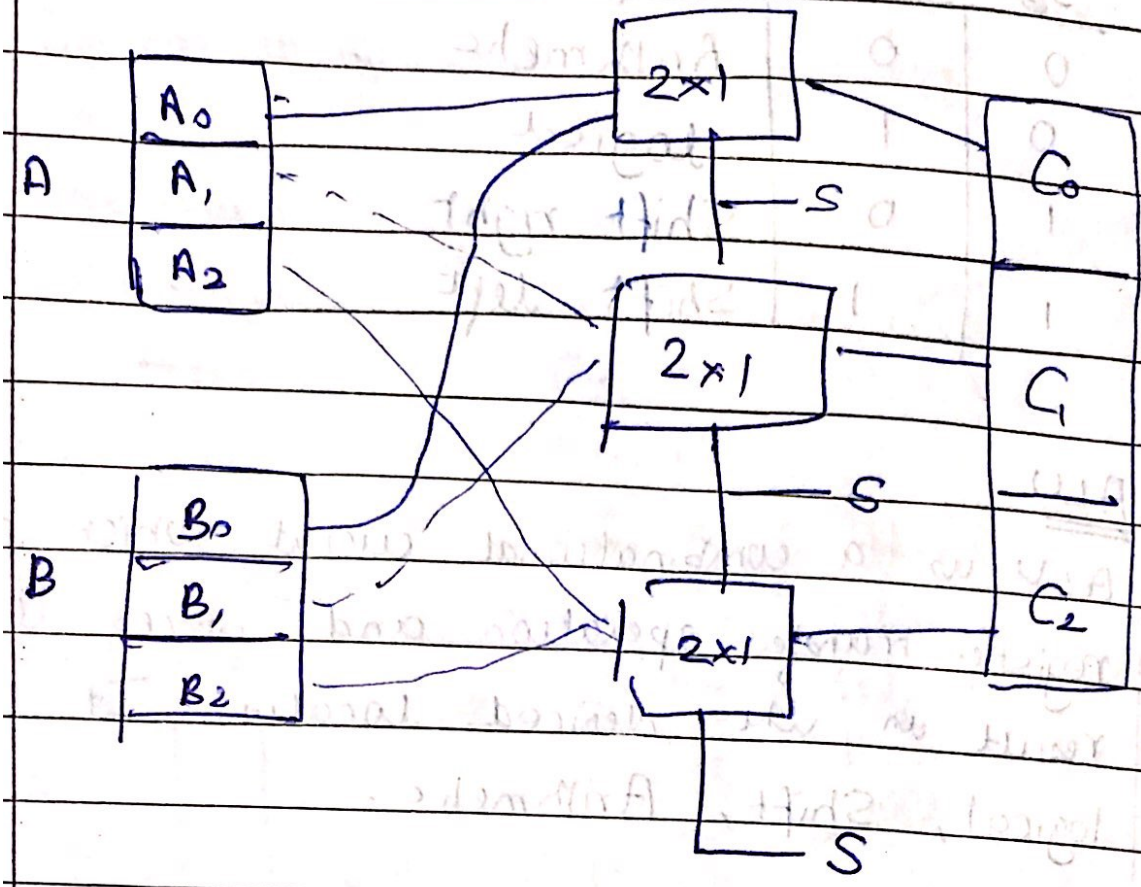
S_3	S_2	operation
0	0	Arithmetic
0	1	logical
1	0	Shift right
1	1	Shift left

ALU

ALU is a combinational circuit which performs register transfer operation and stores the result in the desired location. It performs : logical, Shift, Arithmetic.

ALU performs all the kinds of micro-operations.

Bus : A Bus is a system for transferring information b/w the registers.
Bus system can be constructed using either by multiplexer or a 3-state buffer.



S	$C_0 C_1 C_2$
0	$A_0 A_1 A_2$ (A)
1	$B_0 B_1 B_2$ (B)

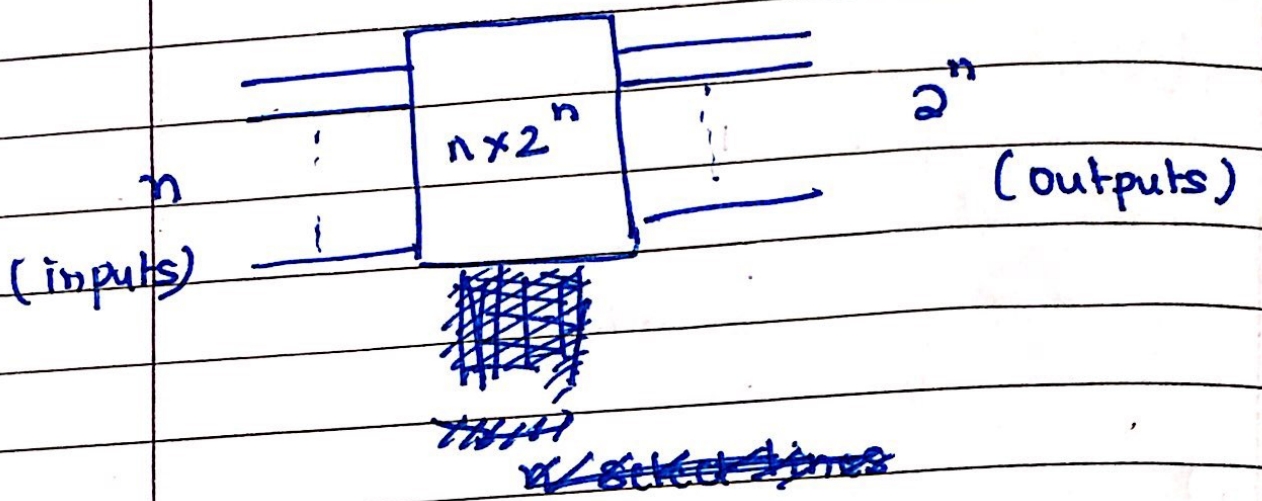
Quiz 3

* Bus System using 3-State Buffer Gates.

$A = A_0 A_1$

$B = B_0 B_1$

$C = C_0 C_1$



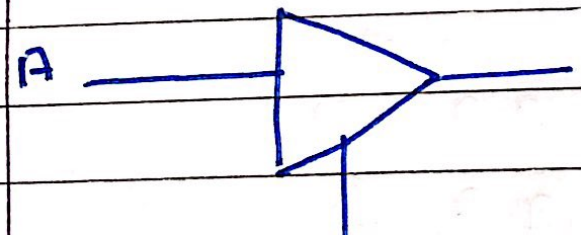
~~select lines~~

{decoder}

No select lines in decoder.

Select lines are only present in multiplexer and demultiplexer.

Three state buffer



Y = A ; C = 1

Y = open ; C = 0

Circuit

Decoder used ?

As only one register we need to use. (Available registers are 2, only register used be should at a particular time).

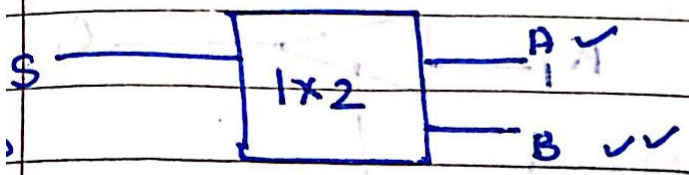
$n \times 2^n$

1×2

2×4

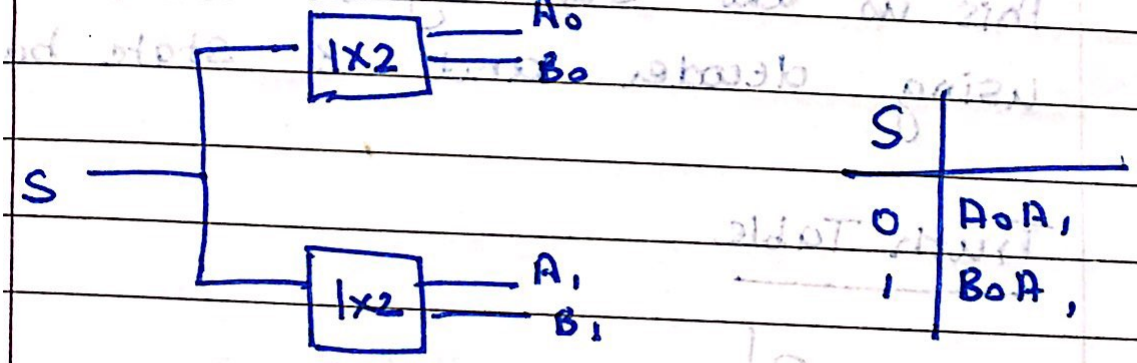
3×8

4×16.



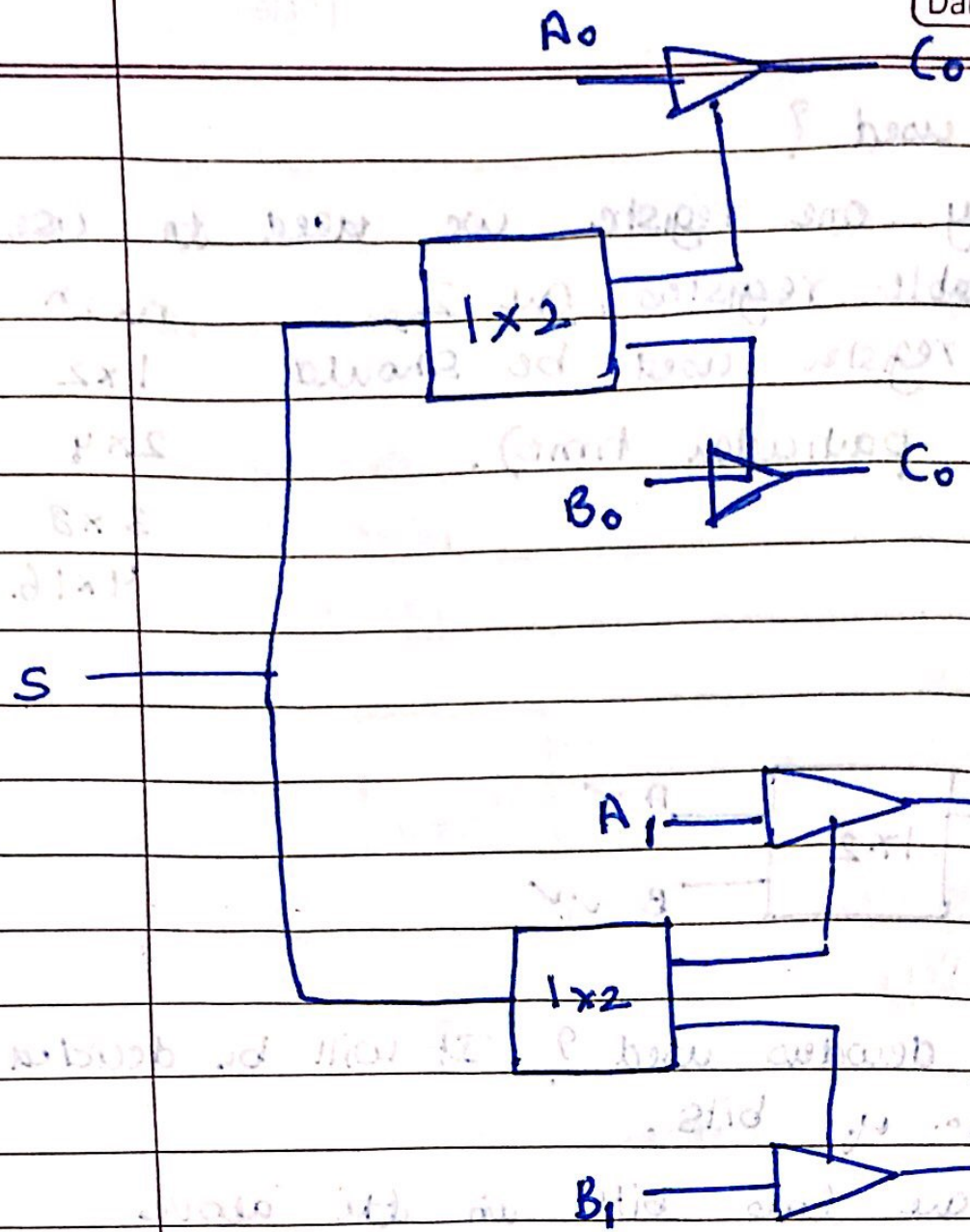
No. of decoders used ? It will be decided by no. of bits.

There are two bits in the above question. Hence two decoders will be used.



This would have been the answer if we had to make without buffer (3 state).

Buffer acts as switch. If C = 1, the input would be equal to output



This is the bus system architecture using decoder and 3 state buffer.

Truth Table

S	Output (C_0C_1)
0	A_0A_1 (A)

Page No.
Date

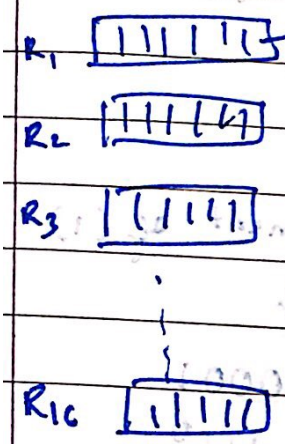
Bus structure for 3bit registers.

$A = A_0 A_1 A_2$

$B = B_0 B_1 B_2$

$C = C_0 C_1 C_2$.

A common bus system is designed for 16 registers of 32 bits each using multiplexers. How many multiplexers are there in the bus and what is the size of each multiplexer.



32 multiplexers required

16x1 size MUX.

$\log_2 16 = 4$

(Select lines).

Generalised form

If there are n multiplexers and K registers

Then, multiplexers no. = n

Size of multiplexer = K x 1

Ques: Design a 4 bit decrementer using four full adders.

$A + \bar{B} + 1$

$= A - B$

$A + \bar{B} + 1$

This is 2's compliment

4-bit can also be written as:

4 + 2's compliment of (6)

$= A_0 A_1 A_2 A_3 - 1$

$= A_0 A_1 A_2 A_3$ + 2's complement of (1)

⇓

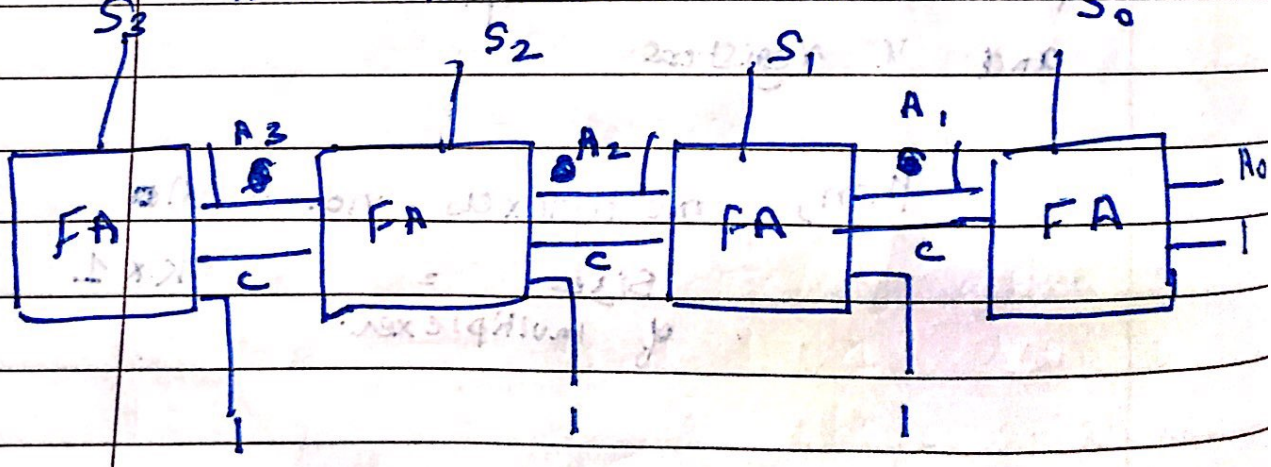
0001

1's ‖ 1110

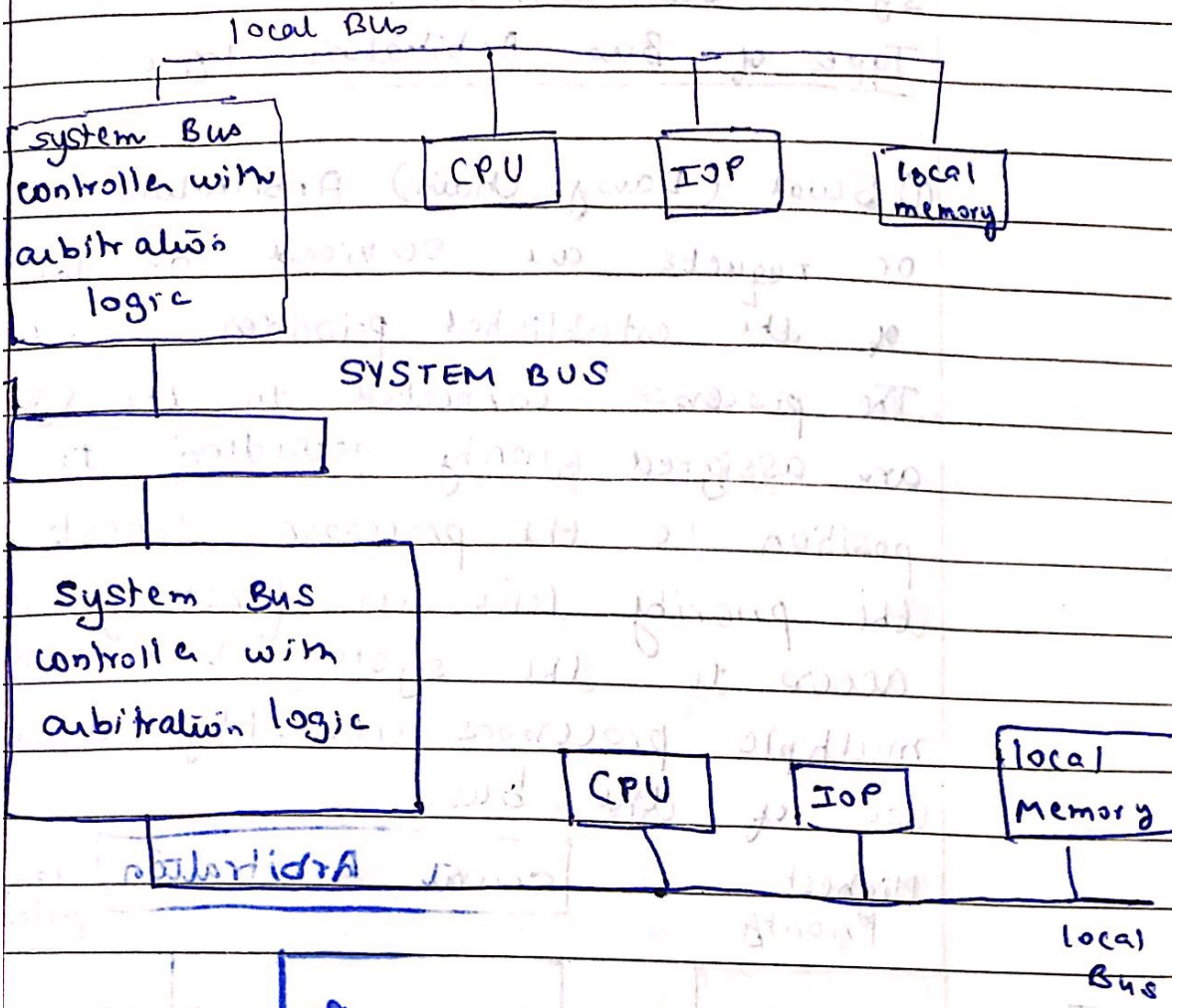
\+ 1

2's compliment 1111

$A - 1 = A_0 A_1 A_2 A_3 + 1111$



Bus Arbitration



Or

In a multiprocessor shown above, the processors request access to share memory with system bus controller (via system bus).

If no processor is utilizing the system bus, the requested processor is granted access to the shared memory.

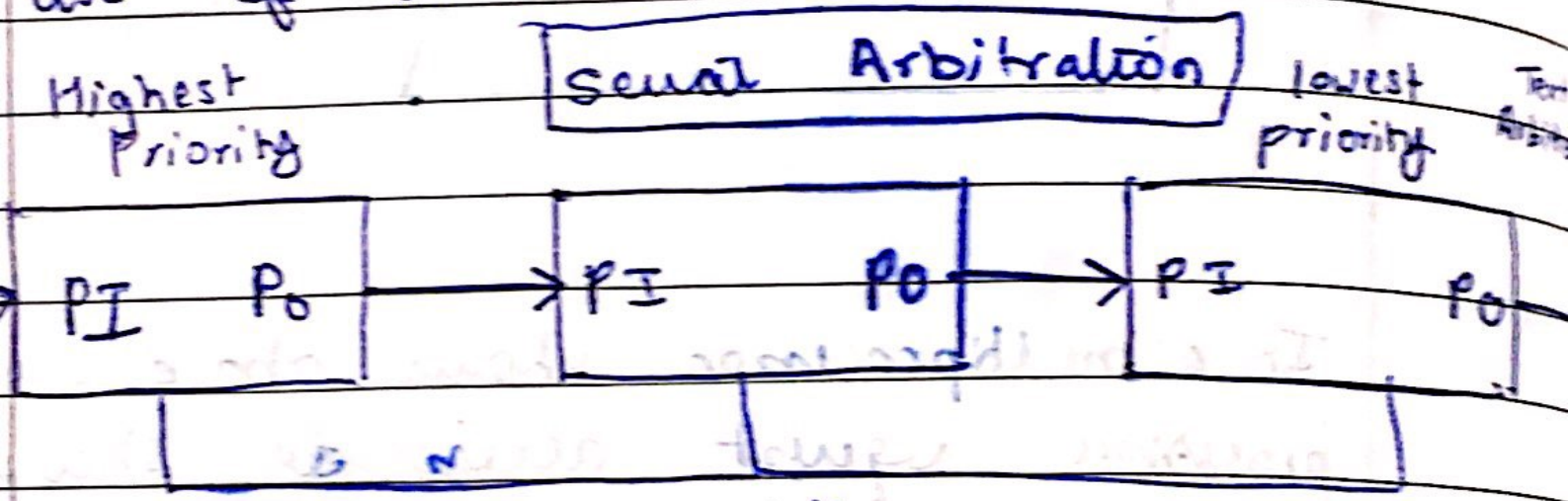
Arbitration is performed if two or more processors are requesting the shared memory via the system bus at the same time.

Arbitration decides which processor gets access to the shared memory and other common resources through system bus.

Arbitration logic is made [illegible] of the system bus controller.

Types of Bus Arbitration logic:

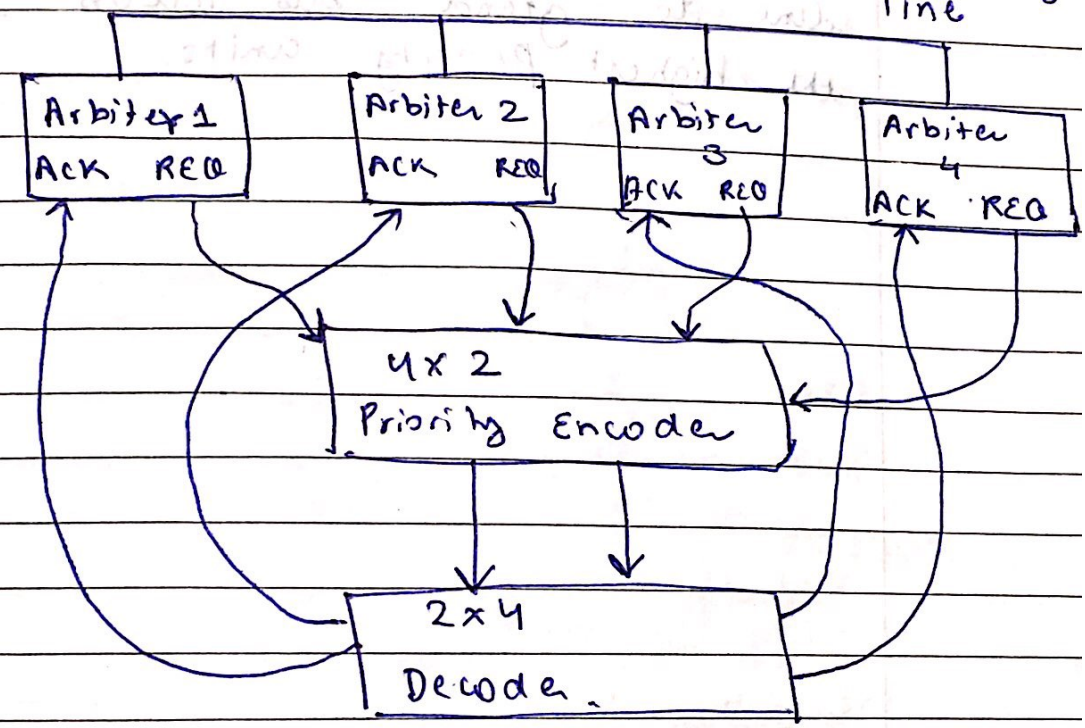
① Serial (Daisy chain) Arbitration: Process or requests are serviced on the basis of the established priorities. The processor connected to the system bus are assigned priority according to their position ie the processor closest to the priority unit is first given access to the system bus when multiple processors currently access use of the bus.



PI of the highest priority unit is

The next in line lower priority arbiter receives a '0' in PI meaning that the bus is currently in use. It has to wait till it can access the system bus.

Parallel Arbitration



Each Bus Arbiter has a bus request output line and a bus acknowledge input line.

The request line is enabled when the processor is requesting access to system bus

The processor takes control of the bus if it its acknowledge input line is enabled.

→ The output of the encoder generates a 2 bit code which represents the highest priority among those requesting the bus.

→ The two bit from the encoder output drives a 2×4 decoder which enables the proper acknowledge line to grand bus access to the highest priority units.

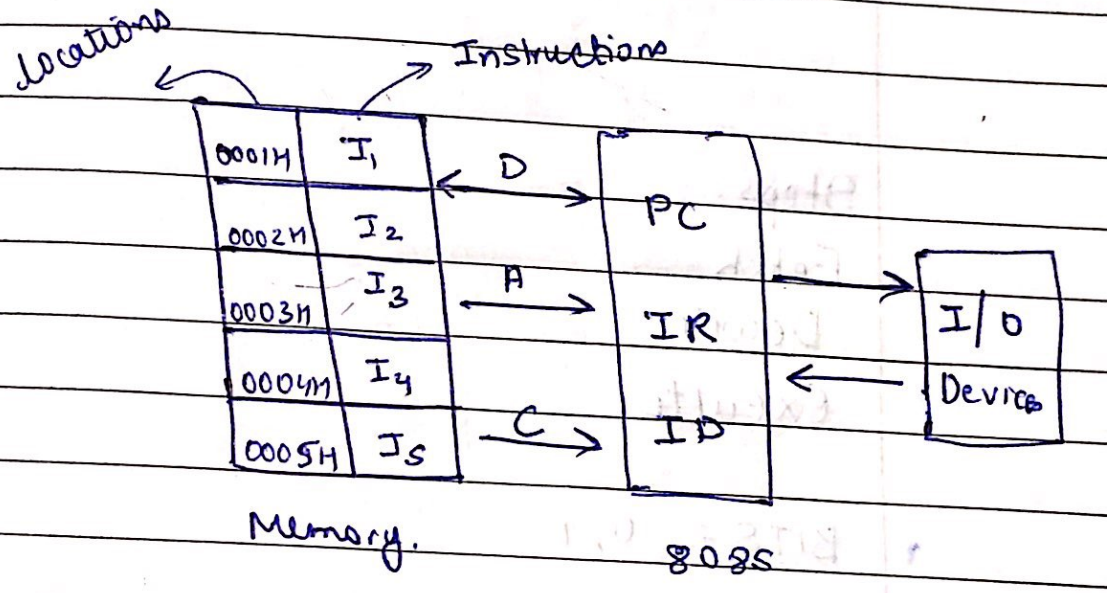
8085 Microprocessor

Input → Processor → Output

Assembly language Program

```
MVI  A, 01H
MVI  B, 02H
ADD  B
STA  9000H
HLT
```

004 (4 bit)
008 (8 bit)
085 (8 bit)
086 (16 bit)



1 line → 0, 1

2 line → 00, 01, 10, 11

⋮

16 lines = 2^{16} = 64 KByte

Address bus identifies the memory location. Data Bus is of 8 bits and Address Bus is of 16 bits.

Steps:

Fetch

Decode

Execute

BITS = 0, 1

NIBBLE = 4-bits

BYTE = 8-Bits

WORD = 16-Bits

Page No.
Date

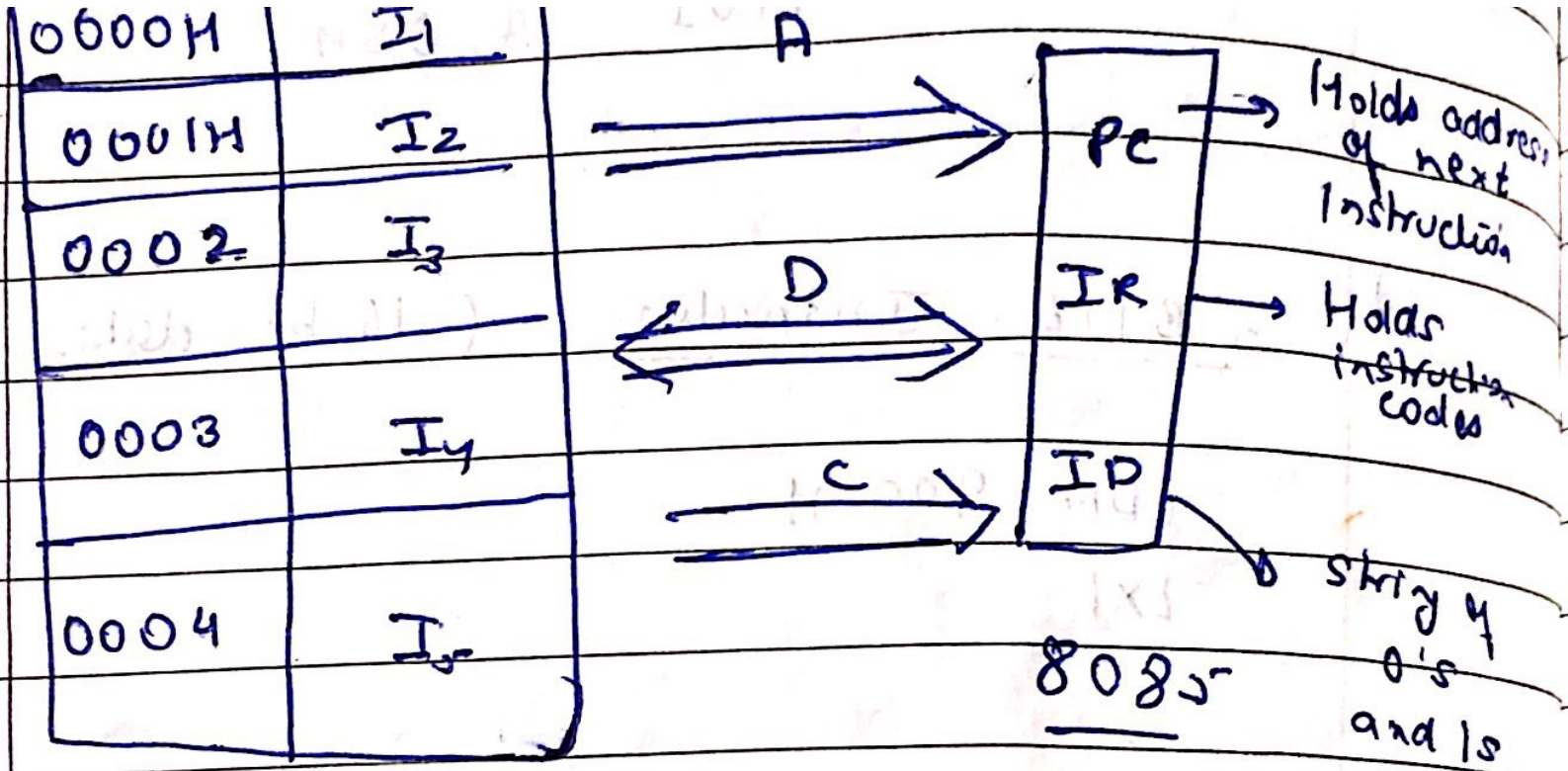
Instruction Size

a) 1 BYTE : ADD B
MOV A,B
SUB B
INR B

b) 2 BYTE : ADI 01H
SUI 02H
MUI A, 05H

c) 3 BYTE Instruction (16 bit data).

LDA 9000H
LXI



① Memory Reference
- Direct Addressing
- Indirect Addressing

② Register Reference

③ I/O Instructions.

Memory reference instructions are those when an operand is needed to carry

Register Reference

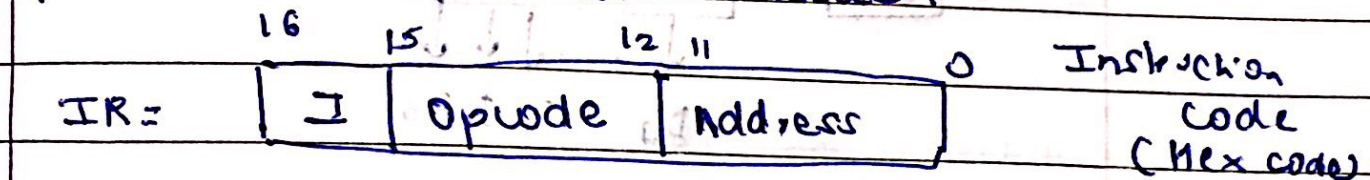
There is no need of data from the memory. The operation is performed on the register only.

CLA: Clear accumulator

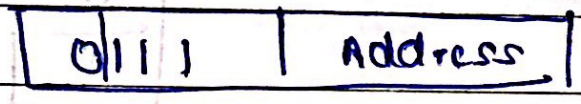
CMA: ~~Clear~~ compliment the result of accumulator.

OUT PORT 1 : There is no need of reference to memory.

fetch, decode and execute: → steps to execute any instruction.



∴ I = 0 (Direct Addressing)
∴ I = 1 (Indirect Addressing) → Memory Reference



= Register Reference

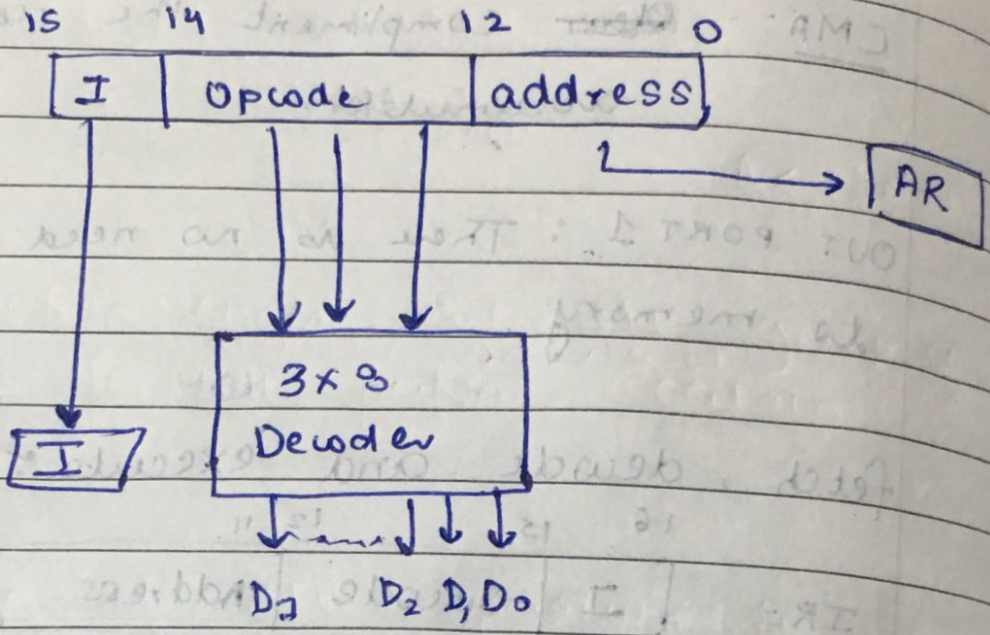
Page No.
Date

Instruction Cycle

Fetch : T_0 : AR ← PC

I_1 : IR ← M[AR]

Decode T_2 :



Determine type of Instruction

T_3 ⟶

I	D_7	Type of Instruction
0	1	Register
1	1	I/O
0	0	Memory reference (Direct)
1	0	Memory reference (Indirect)

r : $D_7 I' T_3$

If $r = 1$ → Type of Instruction

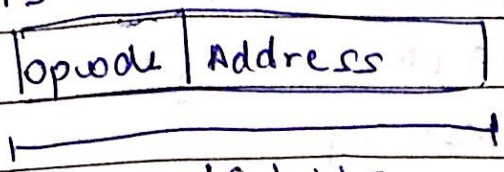
Page No.
Date

Instruction codes

(a) Immediate (operand is provided immediately)

(b) Direct Addressing. eg: ADD B

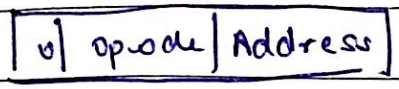
MVI A, 0411.

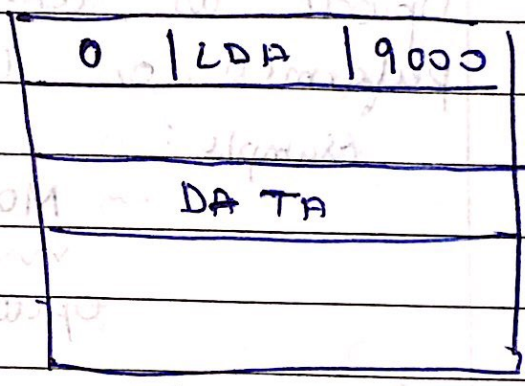


(c) Indirect addressing.

(b) Direct address,

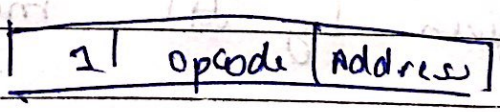
↳ Eg:- LDA 9000H





(c) Indirect address,

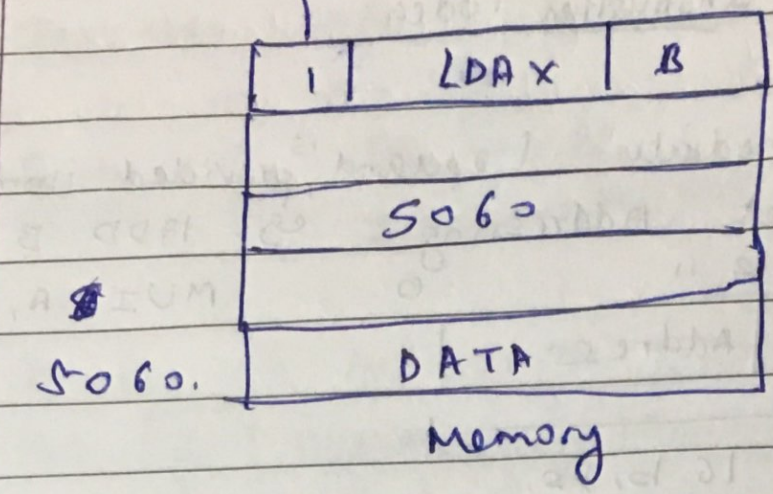
For eg: LDAX B.



LDAX B.

Indirect address, BC = 5060

5060H - DATA



A ← DATA.

Instruction codes tell the computer to perform certain operations. and the group of bits to

Opcode is the operation to be performed. on the operands.

Example:

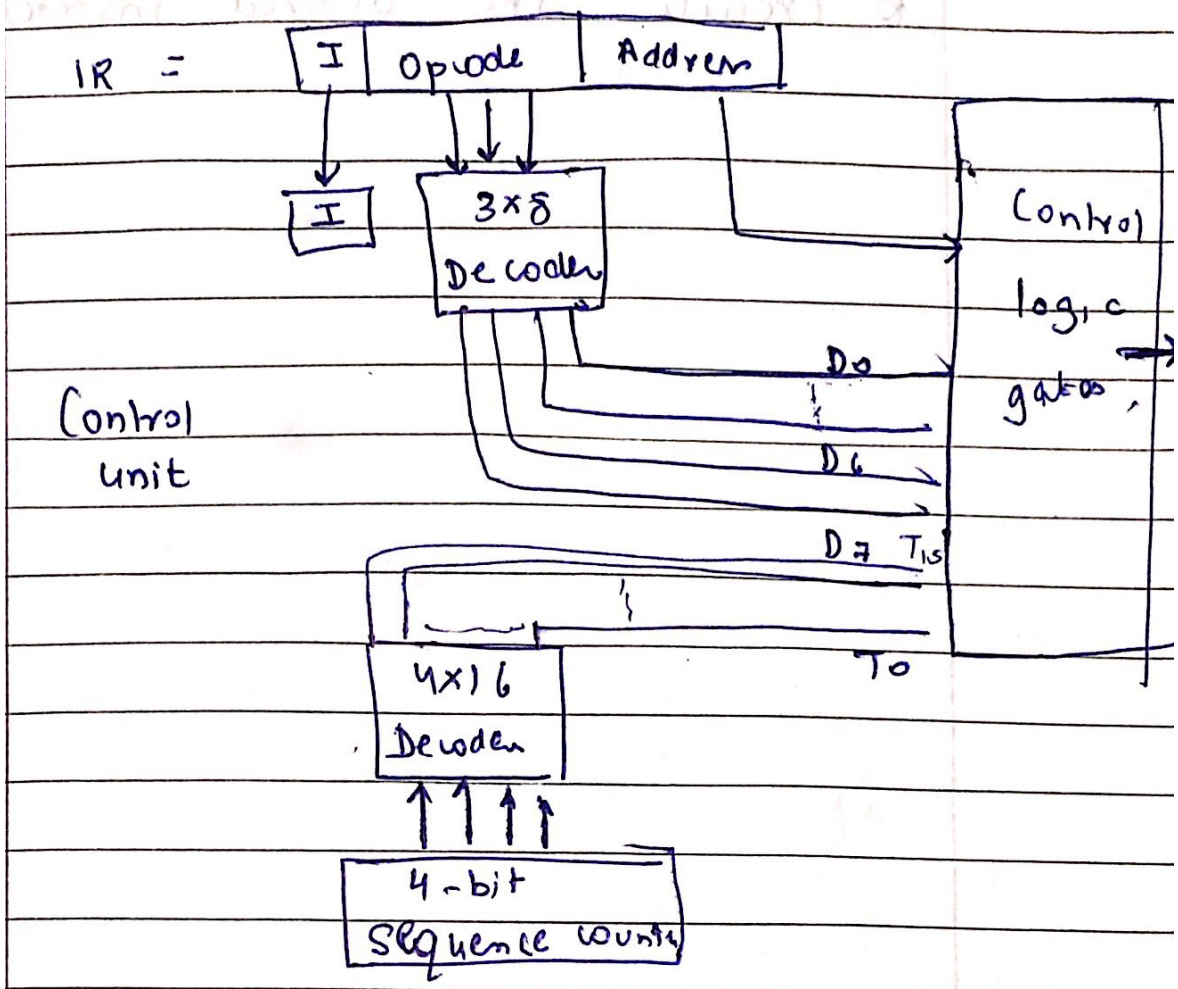
MOV A, B

opcode operands.

Immediate addressing code: address acts the operand itself.
The address of the operand is given by the memory address.

The MSB of instruction code is I in indirect addressing.

LDAX B



The Address of the first instruction to be executed is placed in the program counter through the address bus. The corresponding instruction is selected in the memory and placed in the IR (Instruction Register) through the databus. The instructions in the IR is stored is stored in the form of instruction codes. (hex codes).

The instruction in the IR is decoded using 3×8 decoder.

Page No.

Date

The Control unit generates the control signals to the different blocks of a comp to execute the desired instructions.