1) Define Computer organisation, Computer design and computer architecture

A. Computer organisation :- Computer organisation deals with the structure and behaviour of computer system as seen by the user.

* It deals with the components of a connection in a system computer organisation.

* It tells us how exactly all the unit in the system are arranged and interconnected where as an organisation express the realization of architecture.

* An organisation done on the basis of architecture. Computer organisation deals with low-level design issues.

* Organisation involves physical components (circuit design Adders, signals, peripherals)

Computer Design :- The architectural design of a computer system is concerned.

with the Specification of various functional modules, such a processor and memories and structing them together into a computer system.

Computer Architecture:- Computer Architecture is concerned with the way hardware components are connected together to form a computer system. It acts as interface between hardware and software.

* Computer architecture helps us to understand the functionalities of a system.
* A programmer can view architecture is considered first.
* Computer architecture deals with high level design issues.

2) What is Micro Operation? Write about Register Transfer language.

The operation on data stored in registers are called micro operations. A micro-operation is an elementary operation performed on the information stored in one or more registers. Examples of micro-operations are shift, count, clear and load

Register Transfer language:-

* The symbolic notation used to describe the micro-operation transfer among registers is called RTL.

* The use of symbols instead of narrative explanations provides an organized and concise manner for listening the micro-operation sequences in registers and the control functions that initiate them.

* A register transfer language is a system for expressing in symbolic form the micro-operation sequences among the registers of a digital module.

* It is a convenient tool for describing the internal organisation of digital computers

in concise and precise manner.

LONG ANSWER QUESTIONS:-

3) With the help of examples, explain in detail various types of memory reference instructions.

A *

Symbol	Operation decoder	Symbolic des-criptio
AND	D_0	$AC \leftarrow AC \wedge M[AR]$.
ADD	D_1	$AC \leftarrow AC + M[AR]$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AL$.
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 $PC \leftarrow PC+1$

* The effective address of the instruction is in AR and was placed there during timing signal T_2 when 1=0, or during timing signal T_3 when I=1.

* Memory cycle is assumed to be short enough to complete in a CPU cycle

* The execution of MR instruction starts with T_4

AND to AC

$D_0T_4: DR \leftarrow M[AR]$. Read Operand

$D_0T_5: AC \leftarrow AC \wedge DR, SC \leftarrow 0$ AND with AC

ADD to AC

$D_1T_4: DR \leftarrow M[AR]$. Read Operand

$D_1T_5: AC \leftarrow AC + DR, E \leftarrow Cout, SC \leftarrow 0$ Add to AC and store carry in E

LDA: load to AC

$D_2T_4: DR \leftarrow M[AR]$

$D_2T_5: AC \leftarrow DR, SC \leftarrow 0$.

STA: store AC

$D_3T_4: M[AR] \leftarrow AC, SC \leftarrow 0$.

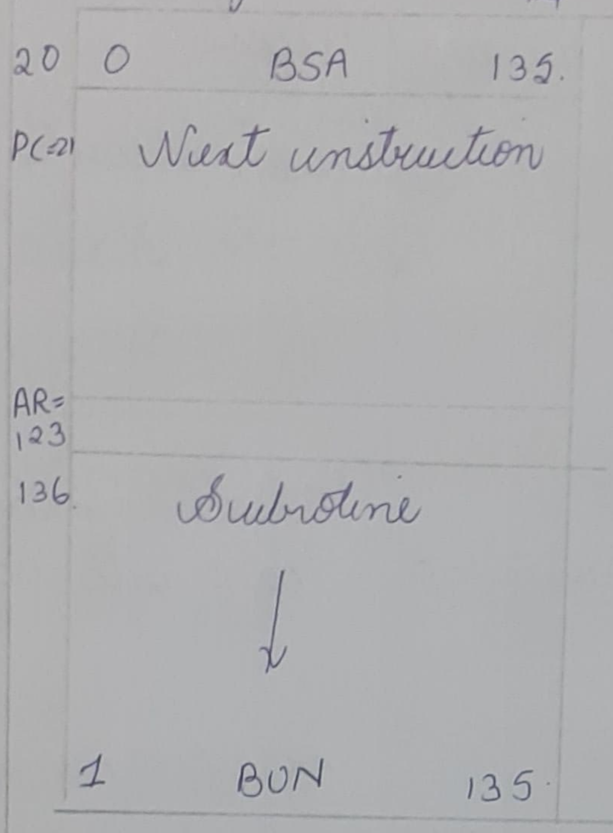
BUN: Branch unconditionally

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

BSA: Branch and save return address

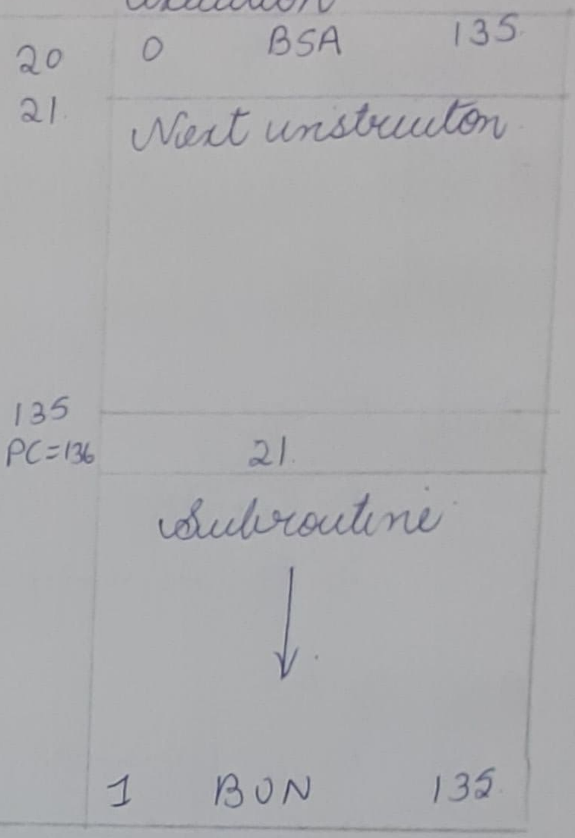
$M[AR] \leftarrow PC, \quad PC \leftarrow AR+1$

Memory, PC, A&R at times T4



Memory

Memory, PC after execution



Memory

BSA:

D_5T_4: M [AR] ← DC, AR ← AR+1.

D_5T_5: PC ← AR, SC ← 0.

ISZ: Increment and Skip if zero.

D_6T_4: DR ← M [AR]

D_6T_5: DR ← DR +1

D_6T_4: M [AR] ← DR, if (DR = 0) then (PC ← PC+1), SC ← 0.

4). Draw and Explain about the Instruction Cycle state diagram.

* In basic computer, a machine instruction is executed in the following cycle.

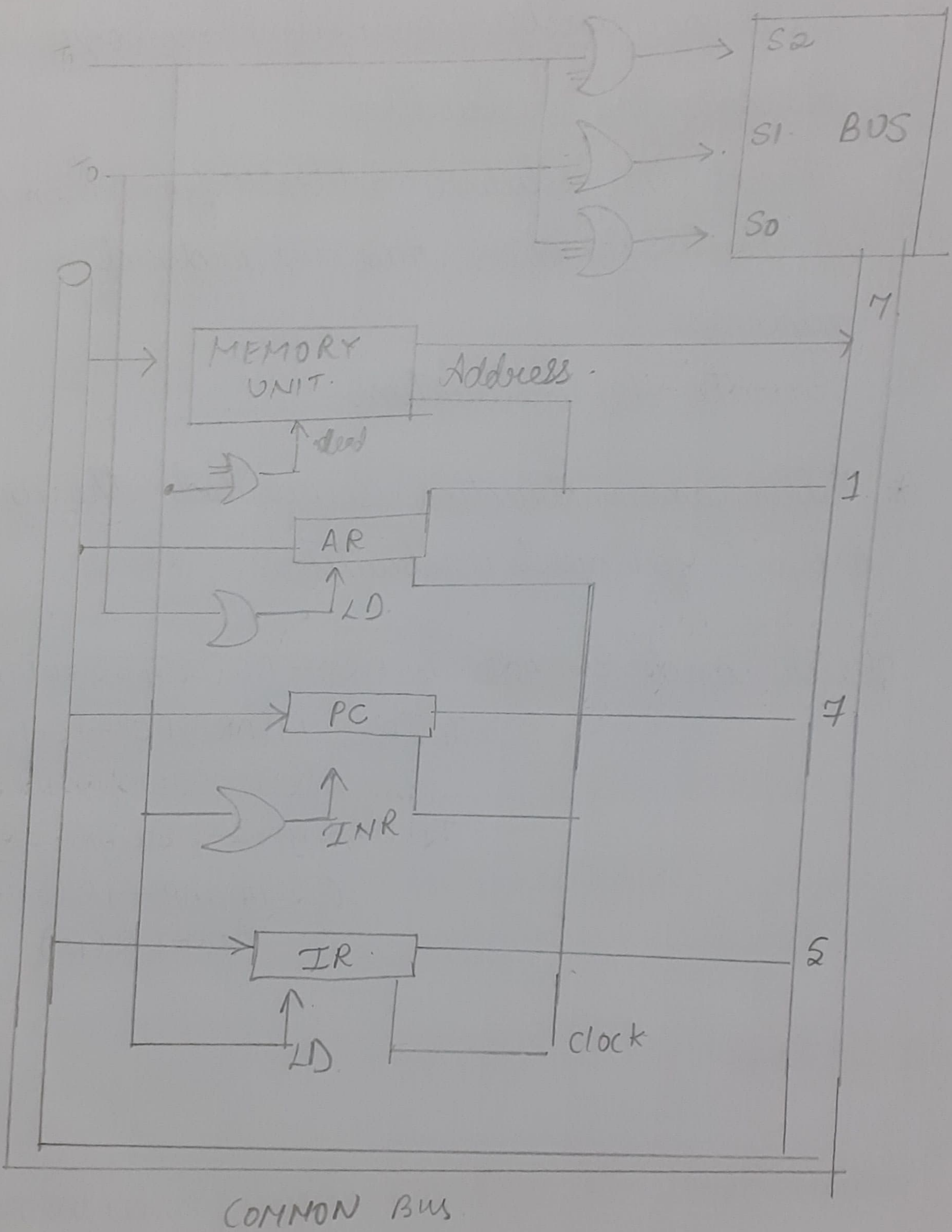
1. Fetch an instruction from memory.
2. Decode the instruction
3. Read the effective address from memory if the instruction has an indirect address.
4. Execute the insitruction

* After an instruction is executed, the cycle Step 1, for next instruction

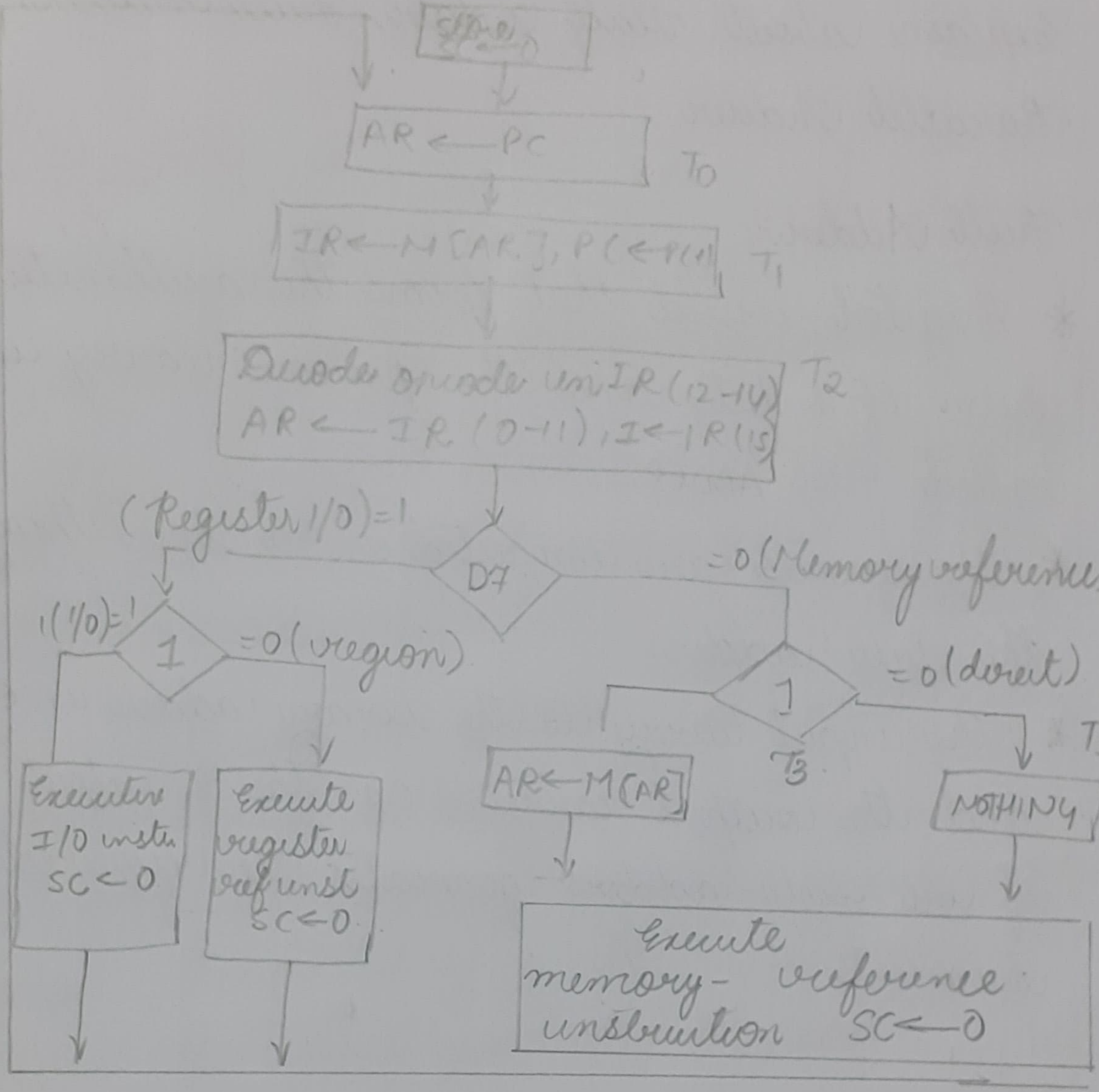
Fetch and Decode: T_0: AR ← PC ($S_0S_1S_2 = 010, T_0 = 1$)

T_1: TR ← M[AR], PC ← PC+1
($S_0S_1S_2 = 111, T_1 = 1$)

T_2: $D_0, \ldots D_7$ ← Decode IR
(12-14), AR ← IR (0-11).
I ← IR (15)


T1
T0
S2
S1
BUS
S0
7
MEMORY UNIT
Address
Read
1
AR
LD
PC
7
INR
IR
5
LD
clock
COMMON BUS

THE TYPE OF INSTRUCTION.



$D'_7 I T_3$: AR ← M[AR].

$D'_7 I' T_3$: Nothing.

$D_7 I' T_3$: Execute a register reference instr

$D_7 I T_3$: Execute an input-output instruction

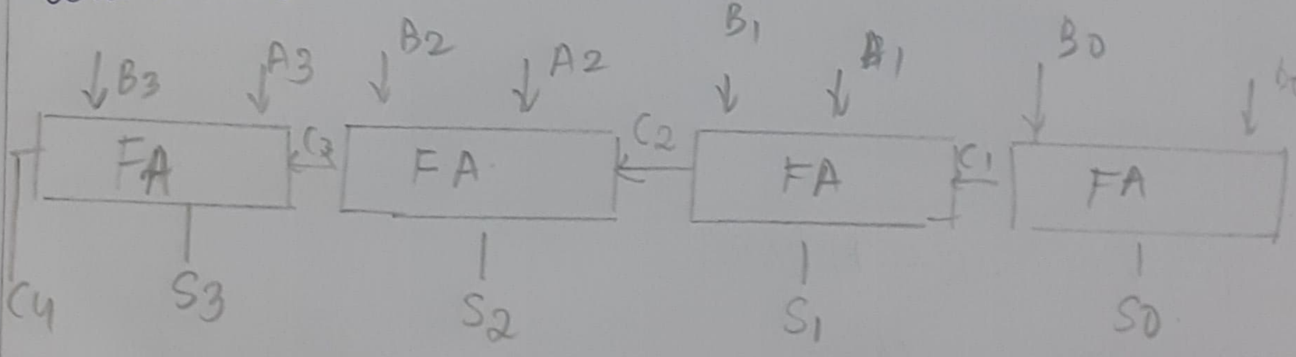
5) With the help of block diagram, Explain about Half Adder, Full Adder, Parallel Adder.

A:- Full Adder:-

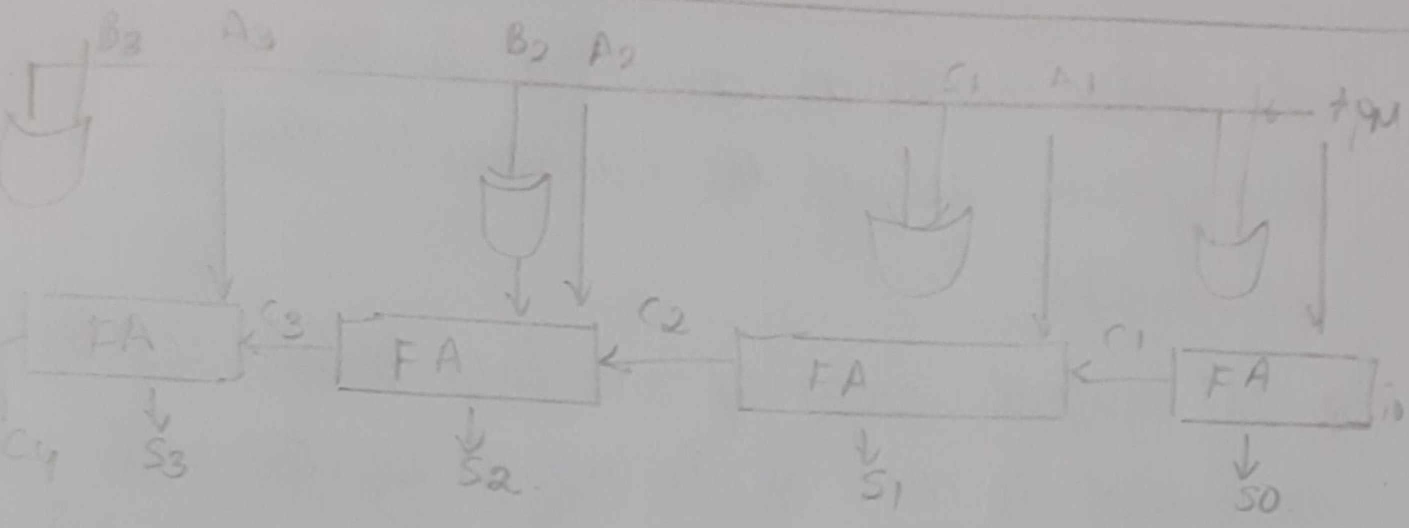
* Digital circuit that forms the arithmetic sum of 2 bits and the previous carry is called FULL ADDER.

* The carries are connected in a chain through the full-adders.

* The input carry to the binary adder in C_0 and the output carry is C_4. The S output of the full-adders generates the required sum bits.



* The addition and subtraction operations can be combined into one common circuit by including an exclusive-OR gate with each ful
adder



4-bit adder Subtractor

* The mode input M controls the operation when $M=0$, the circuit is an adder and when $M=1$, the circuit seems a subtractor.

* Each exclusive-OR gate receives input M and one of the inputs of B

* When $M=0$, we have BxCOR0 = B. The full adders receive the value of B, the input carry is 0 and the circuit performs A plus B.

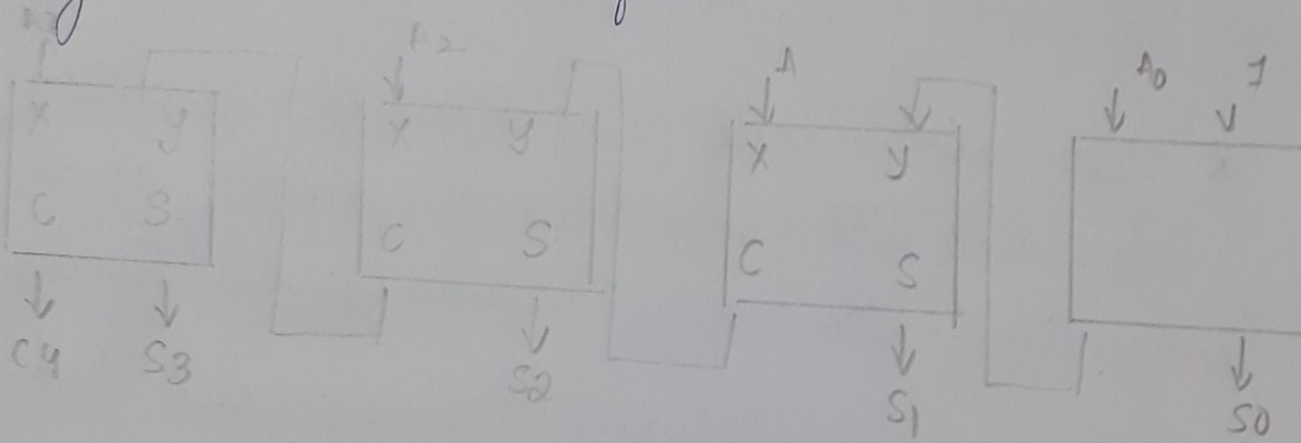
* When $M=1$, we have BXOR1 $= B'$ ie $C_0 = 1$.

Half Adder:-

* One of the inputs to the least significant half adder (MA) is connected to logic 1 ie

the other input is connected to the least significant bit of the number to be incremented.

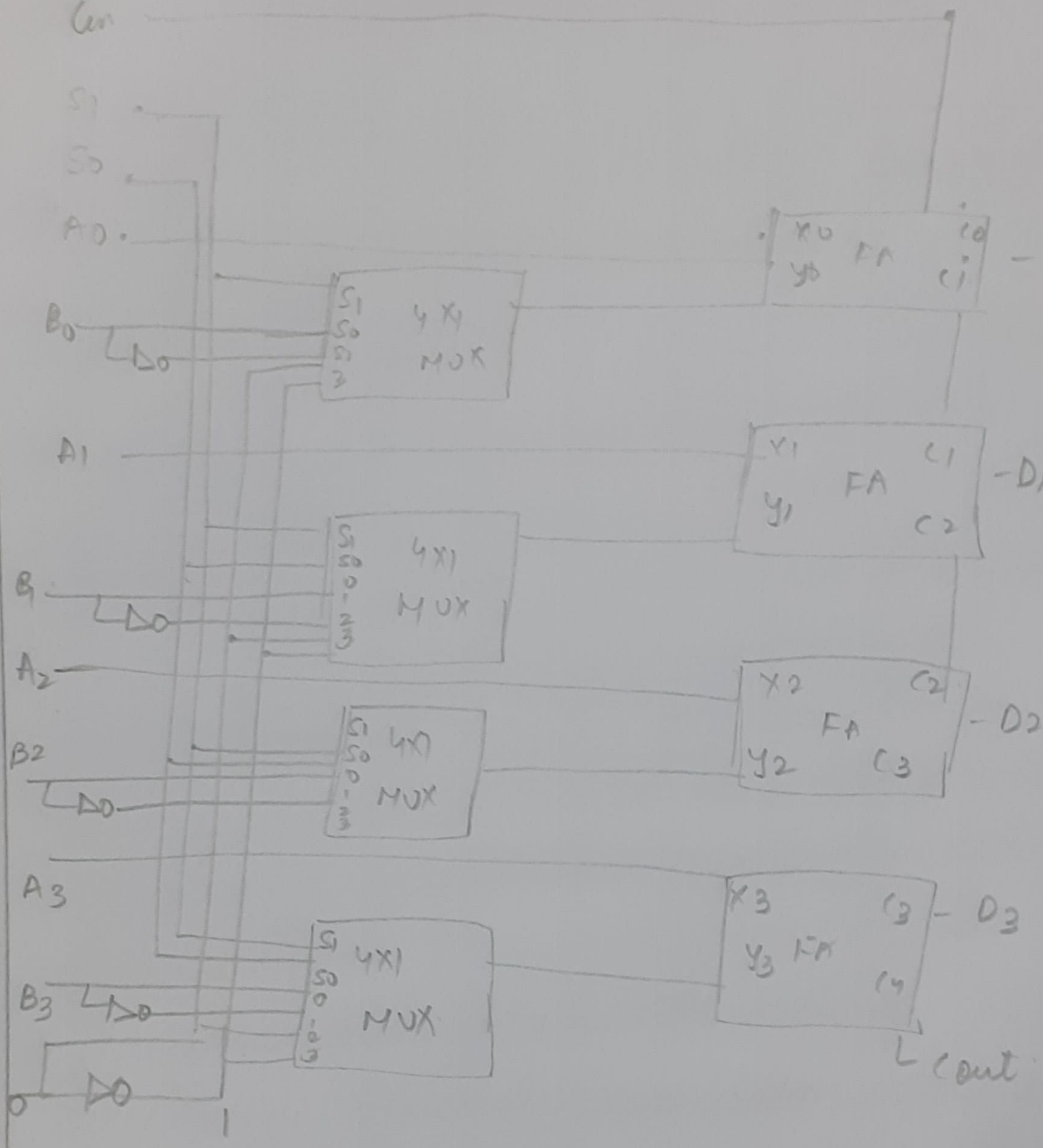
* The output carry of one half-adder is connected to one of the inputs of the next higher ~~level~~ order half adder.



* The circuit can be extended to an n-bit binary incriater by extending diagrams to include n half-address

Parallel Adder:-

* The basic component of an arithmetic circuit is the parallel adder.


Cin
S1
S0
A0
B0
A1
B1
A2
B2
A3
B3
0
1
S1
S0
0
1
2
3
4X1
MUX
X0
Y0
FA
C0
C1
D0
X1
Y1
FA
C1
C2
D1
X2
Y2
FA
C2
C3
D2
X3
Y3
FA
C3
C4
D3
Cout

—X—

END

1). Discuss in Short about Signed 1's complement & 2's complement representation.

1's complement.

This is a simple algorithm to convert a binary number into 1's complement. To get 1's complement of a binary number, simply invert the given number.

2's complement.

There is a simple algorithm to convert a binary no. into 2's complement ~~to~~. To get 2's complement of a binary number, simply invert the given no. & add 1 to the least significant bit (LSB) of given result

2). Write a Short on floating point representation of deaned number.

This representation does not deserve a Specific no. of bits for the integer or ~~fast~~ fractional part. Instead it reverses a certain no. of bits for the no. and a certain no. of bits to say where within that no. the decimal place sits. The floating no. representation of a no has two part. The first part represents a ~~Signal~~ Signed point no. called mantissa may be fraction or an integer. Floating point is always interpreted to represent a number in the following

form $M \times r^e$

Only the mantissa m & the exponent e are physically represented in the registers. A floating point no is said to be normalised

If most Significant digit of mantissa is 1

n 0

Sign bit	Exponent.	Mantissa

Baised form

Q3). Explain IEEE floating representation in single precission and double precision format with an example.

Single Precision.

32 bits

Sign	Exponent	Mantissa
1 bits	8 bits	23 bits

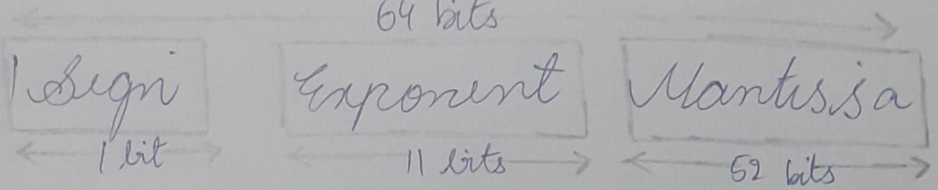
Sign : 1 (31st bit).

passed exponent : 8 (20-23).

Normalised Mantissa : 23 (22-0).

Bias : 127.

Double Precision



Sign = 1 (. bit).

Biased Exponent : 11 (62-65).

Normalised Mantissa : ~~23 (22-0)~~ 53 (51-0)

Bias : 1023.

Example :

85.125.

85 = 1010101 ; 0.125 = 001.

85.125 = 1010101.001.

= 1.010101001 × 2^6

Sign = 0

1) Single Precision

Biased exponent 127+6 = 133.

133 = 10000101.

Normalised Mantissa = 010101001

we will add 0's to complete the 23 bits

2) Double Precision

biased exponent 1023+6 = 1029

1024 = 10000000101

Normal Mantisa = 010101001

we will add 0's to complete the 52' bits

The IEEE 754 double precision is = 0

10000000101 01010100100000000000000

Explain Booths Multiplication algorithm with an example using necessary diagrams

Booth algorithm is a multiplication algorithm that multiplies two signed binary no in 2's compliment notation

PROCEDURE :-

1) let M is the multiplicand

2) let Q is the multiplier

3) Consider a 1-bit register Q_{-1} & initialize it to 0

4) Consider a register A & initialize it to 0.

M=6=0110

Q=2=0010 (Q_3, Q_2, Q, Q_0).

Both algorithm calulate the product in n-steps where n is the no of bits used to represent the no.

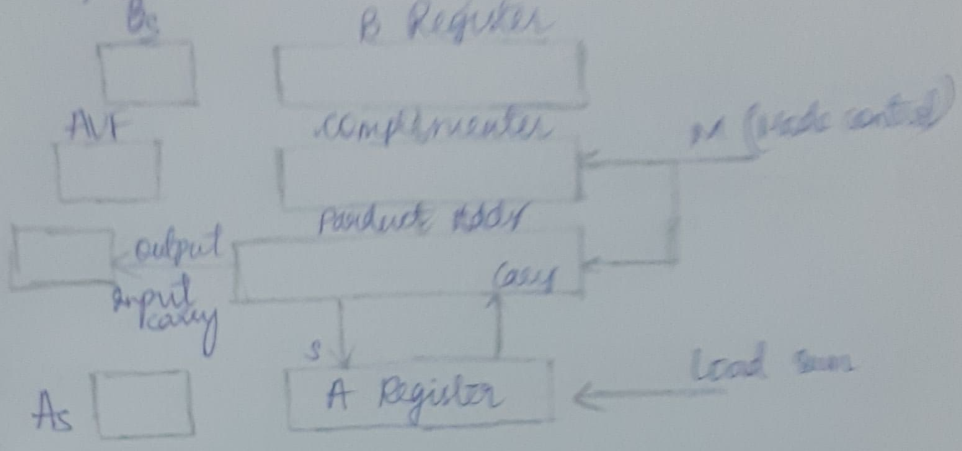
Q5) Explain about decimal subtraction operating using flow chart & hardware configuration with an example.

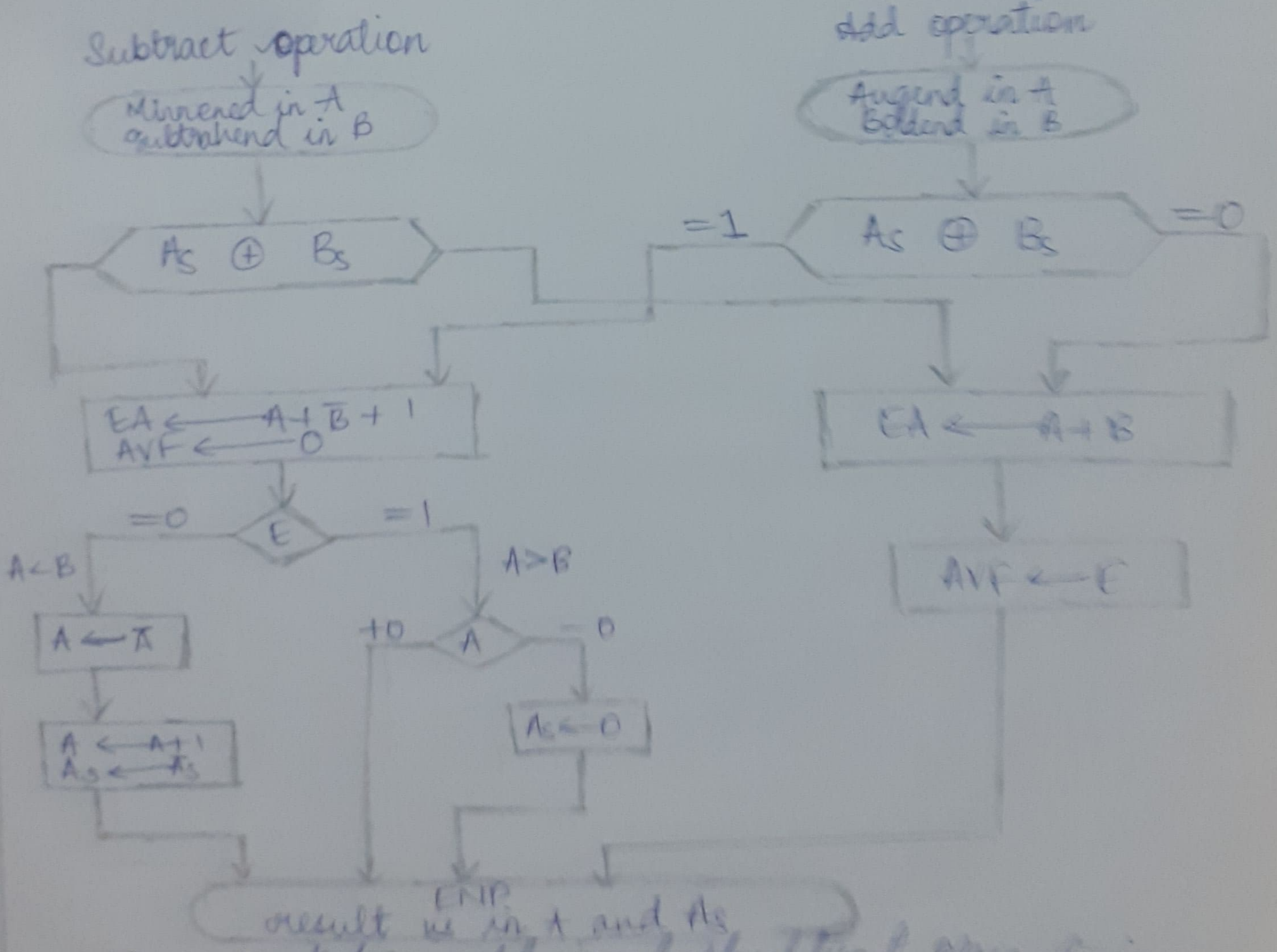
We ~~a~~ designated the magnitude of 2 no's. by A & B where the signed no's are added or subtracted. We find that there are 8 diff conditions two consider, dependence on the sign of no's & the operations performed. The algorithm for ~~condition~~ addition & subtraction are derived from the Table & can be started.

Operation	Add	Subrat Magnitude		
	Magnitude	where A>B.	when A< B	when A=B
(+A) + (B)	+(A+B).	-(A+B)	-(B-A)	+(A-B)
(+A) + (-B).				
(-A) + (+B).		-(A-B)	+(B-A)	+(A-B
(-A)+(-B)	-(A+B)			〃
(+A) - (+B).		+(A-B)	-(B-A)	+(A-B)
(+A)-(-B)	-(A+B).			
(-A)-(+B)	〃	-(A-B	+(B-A).	+(A-B)
(-A)-(-B)				

Subtraction: A-B : A: m[illegible], B: subtrahend

Hardware umpliminitation





flow chart for add and subtract operation