COMPUTER ARCHITECTURE paper - IV

Assignment - I

1. Explain about fixed point and floating point Representation.

Fixed point Represention

This represention has fixed number of bits for integer part and for fractional part. for example if given fixed point represention is 1111, FFFF, they you can store minimum value is 0000.00001 and maximum value is 9999.9999. There are three parts of a fixed point number representation. the sign field, integer field and fractional field.

Usigned fixed point	Integral	Fraction	
Signed fixed point	sign	integer	fraction

We can represent these number using

⇒ signed representation: range from $-(2^{(k-1)}-1)$ to $2^{(k-1)}-1)$ for k bits

⇒ 1's complement representation: range from $-(2^{(k-1)}-1)$ to $2^{(k-1)}-1)$ for k bits

⇒ 2's complementation representation: range from $-(2^{(k-1)})$ to $(2^{(k-1)}-1)$ for k bits

2's complementation representation is prefered in computer system because of unambiguous property and easier for arithemetic operations

example: Assume number is using 32-bit format which reserve 1 bit for the sign 15 bits for the integer part and 16 bits for the fractional part.

Then -43.625 is represented as following

1	000000000101011	1010000000000000
Sign bit	integer part	Fractional part

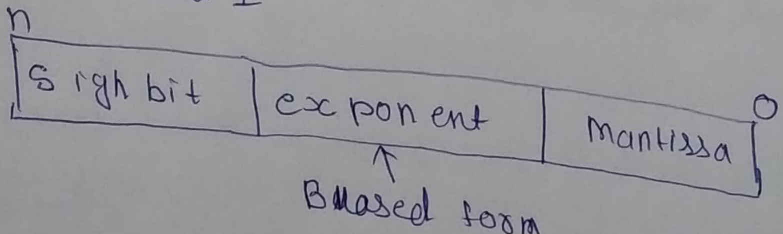
where, 0 is used to represent + and 1 is used to represent, 00 0000000000101011 is 15 bit binary value for decimal 43 and 1010000000000000 is 16 bit binary value for fractional 0.625

Floating point Represention

This represention doesnot reserve a specific number of bits for the integer part or the fractional part. Instead it reverse a certain number of bits for the number and certain number of bits to say where within that number the decimal place sits.

The floating number representation of a number has two part the first part represents a signed fixed point number called mantissa.. flating point is always interpreted to represent a number in the following Mx re

only the mantissa m and the exponent e are physically represented in the register. A floating-point binary number is represented in a similar mannar except that is uses base 2 for the exponent. A floating point number is said to be normalized if the most significant digit of the mantissa 1



So, actual number is $(-1)^s (1+m) \times 2^{(e-Bias)}$, where S is the sign bit, m is the mantissa, e is the exponent value, and Bias is the bias number.

Note that signed integers and exponent are representeed by either sign rebreontation or one's combleement reprosentation, or two's complement reprosentation.

The floating point reprosentation is more flexible. Any non-zero number can be reprosented in the normalized from of $\pm (1.b_1 b_2 b_3) 2 \times 2^n$ This is normalized from of a number x.

2. What are CPU Registers? Explain Them.

CPU registers are small fast storage location within the centeral pro'cessing unit (CPU) of a computer They are used to hold data that the CPU needs to acces quickly while performaning operations. Here som common type of CPU registers and their function.

Accumulator (ACC) used to store Intermediate arithemetic and logic results.

program counter (PC) Holds the addreses of the next instruction to be executed.

Instruction Register (IR) Contains the current instruction being executed

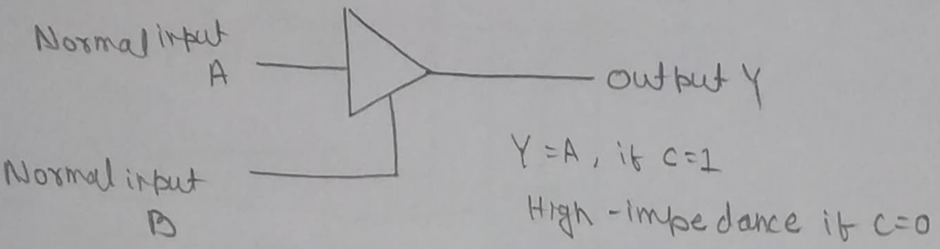
Memory Address Register (MAR) Holds the memory address of data that needs to be accessed.

Memory data Register (MDR) Store the data being transfered to or from the memory location pointed to by the MAR

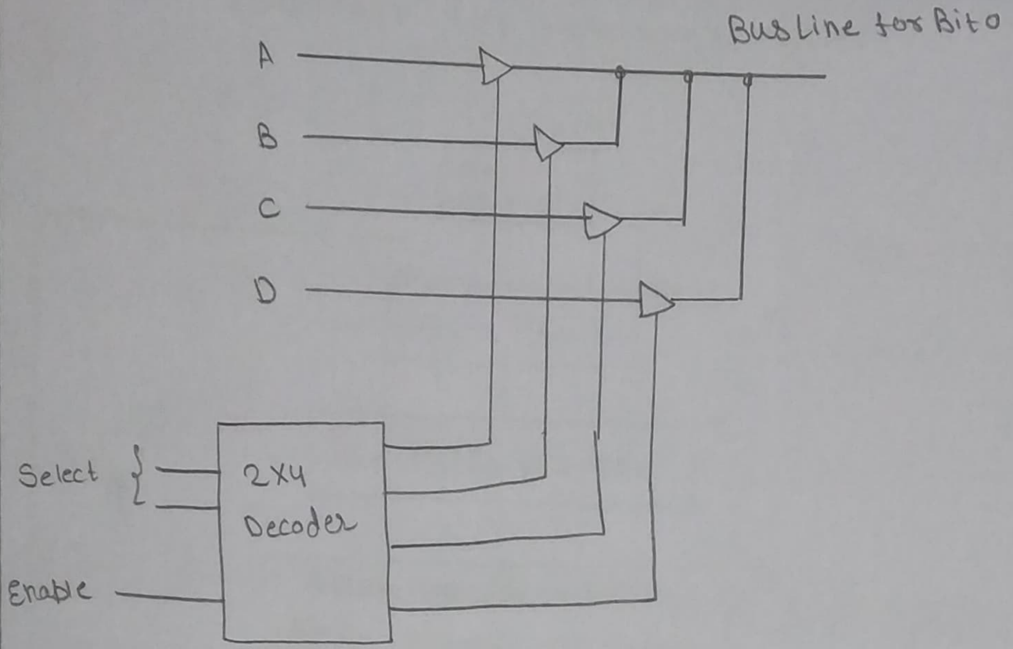
Status Register: Holds flags that indicate the status of the CPU, such as zero carry, overflow and sign flags

3. Construct a Bus System for four Registers using three state Bus Buffers.

A bus system can also be constructed with three-state gates instead of multiplexers. The three state gate is a digital circuit that exhibits three States. Two of the states are signals, equivalent to logic 1 and 0 as in a conventional gate the third state is high-impedance state. Three gats may perform any conventional logic, such as AND or NAND. However the one most commonly used in the design of a bus system is the buffer gate.



It is distinguished from a normal buffer by having both a normal input and a control input. The control input determines the output state. when the control input is equal to 1, the output is enabled and the gate behaves like any conventional buffer; with the output equal to the normal input. when the control input is 0, the output is disabled and the gates goes to a high-impedance state, regardless of the value in the normal input. Because of this feature, a large number of three-state gate output can be connected with wires to form a common bus line without endangering loading effects.

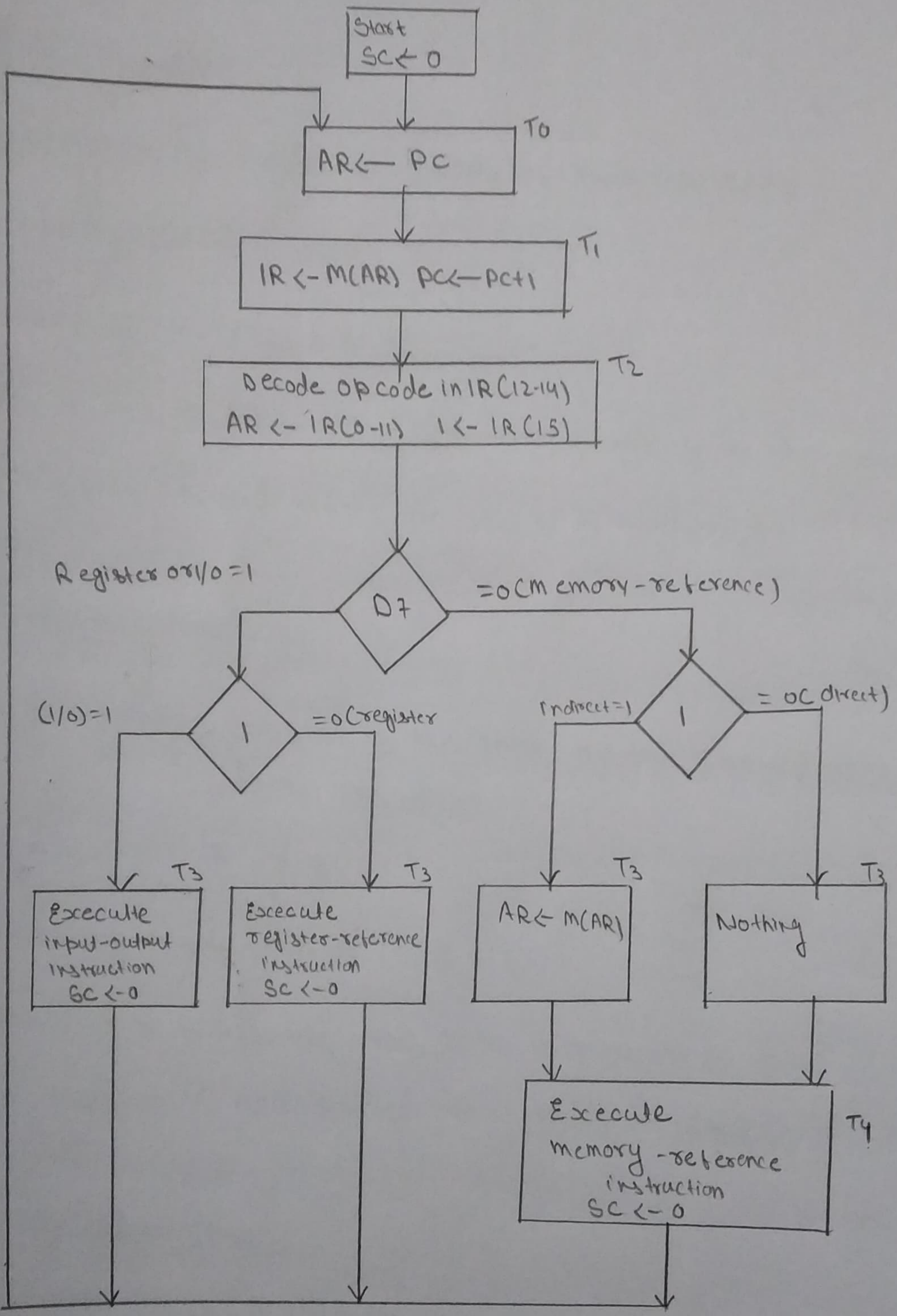


The Construction of a bus system with three state buffers is demonstrated in above figure. The outputs of forur buffers are connected together to form a single bus line. The control inputs to the buffers determine which of the four normal inputs will commu-nicate. with the bus line. No more than one buffer may be in the active state at any given time.

One way to ensure that no more than one Contool input is active at any given time is to use a decoder, as shown in the diagram when theenable input of the decoder is 0, all of its forur output are 0.

4

Draw a flowchart for instruction cycle.



Start SC ← 0
T0
AR ← PC
T1
IR ← M(AR) PC ← PC+1
T2
Decode op code in IR(12-14) AR ← IR(0-11) I ← IR(15)
D7
Register or I/O = 1
= 0 (Memory-reference)
I
(I/O) = 1
= 0 (register
Indirect = 1
= 0 (direct)
T3
Execute input-output instruction SC ← 0
Execute register-reference instruction SC ← 0
AR ← M(AR)
Nothing
T4
Execute memory-reference instruction SC ← 0

5

Write Arithemetic and Logic operations.

Arithemetic operations

Addition (ADD)

Discription ⇒ Adds the values of two operands

example ⇒ 'A = A+B'

Assembly ⇒ 'ADD A, B'

operation ⇒ Adds the value in register B to the value in register A, and stores the result in register A.

Substraction (SUB)

Discription ⇒ Substracts the value of the second operand from the first.

Example ⇒ 'A = A-B'

Assembly instruction ⇒ 'SUB A, B'

operation ⇒ Substracts the value in register B from the value in register A, and stores the result in register A

multiplication (MUL)

Discription ⇒ Multiples two operands

example ⇒ 'A = A*B'

Assembly instruction ⇒ 'MUL A, B'

operation ⇒ multiplies the value in register A by the value in register B. and stores the result in register A.

Division (DIV)

Description ⇒ Divides the first operand by the second

example ⇒ 'A = A/B'

Assembly instruction ⇒ 'DIV A.B'

operation ⇒ Divides the value in register A by the value in register B. and stores the result in register A.

Increment (INC)

Description ⇒ increase the value of an operand by one

example ⇒ 'A = A+1'

Assembly instruction ⇒ 'INC A'

operation ⇒ Adds 1 to the value in register A

Decrement (DEC)

Description ⇒ Decrease the value of an operand by one.

example ⇒ 'A = A-1'

Assembly instruction ⇒ 'DEC A'

operation ⇒ substract 1 from the value in register A.

Logic operations

AND

Description ⇒ performs a bitwise AND operation on two operands

example ⇒ 'A = A ANB'

Assembly instruction ⇒ 'AND A, B'

operation ⇒ performs a bitwise AND between the values in registers A and B. and stores the result in register A

OR

Description ⇒ performs a bitwise OR operation on two operands

example ⇒ 'A = A OR B'

Assembly instruction ⇒ 'OR A, B'

operation ⇒ performs a bitwise OR between the value in registers A and B and stores the result in register A

NOT

Description ⇒ performs a bitwise NOT operation on an operand

example ⇒ 'A=NOT A'

assembly instruction ⇒ 'NOT A'

operation ⇒ Inverts all bits in the value of register A

COMPUTER ARCHITECTURE Paper- IV

Assignment - II

1. Explain about the Design ~~of~~ of Micro programme Seauencer.

The basic components of a micro programmed control unit are the control memeory and the circuit that Select the ~~x~~ next address. The address selection part is called a microprogram seauencer.

There are two multiplexer in the circuit

⇒ The first multiplexer selects an address from one of four sources and routes it into control addressregister CAR.

⇒ The second multiplexer test the value of a selected status bit and the result of the test is applied to an input logic circuit.

The output from CAR provides the address for the control memory. The content of CAR is incremented and applied to one of the multiplexer inputs and to the subroutine register SBR.

The other three inputs to multiplexer come from

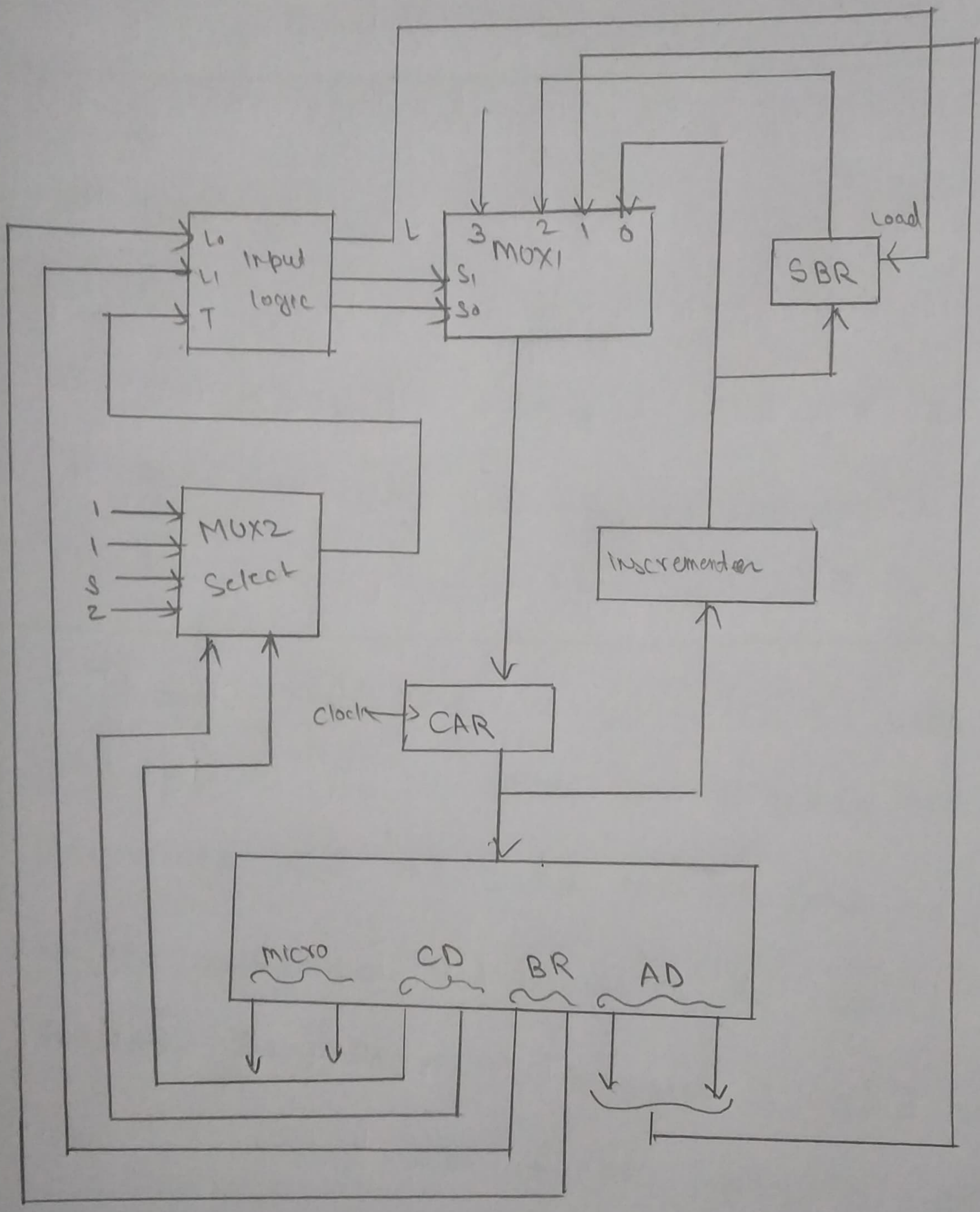
(i) The address field of the present microinstruction

(ii) From the out of SBR

(iii) from an external source that maps the instruction.

The CD (condition) field of the microinstruction selects one of the status bits in the second multiplexer If the bit selected is equal to 1, the T variable is equal to 1. Otherwise it is equal to 0 The T value together with two bits from the BR (branch) field goes to an input logic circuit. The input logic in a particular sequencer will determine the type of operation that are available in the unit

The input logic circuit in below figure has three input I_0 I_1 and T and three output S_0, S_1 and L, variable S_0 and S_1 select one of the source addresses for CAR. variable L enable the load input in SBR The binary values of Selection variables determine the path in the multiplexer ~~for~~ example with $S_1, S_0 = 10$ multiplexer input number 2 is selected and establishes transfer path from SBR to CAR



micro program sequencer for a controlmemory

The truth table for the input logic circuit is show in table below

P.T.O

BR Field	Input I_1 I_0 T	MUX 1 S_1 S_0	Load SBR
0 0	0 0 0	0 0	0
0 0	0 0 1	0 1	0
0 1	0 1 0	0 0	0
0 1	0 1 1	a ϕ	1
1 0	1 0 X	1 0	0
1 1	1 1 X	1 1	0

Input I_1 and I_0 are identical to the bit values in the BR field. The bit values for S_1 and S_0 are determined from the stated function and the path in the multiplexer that establishes the required transfer. The subroutine register is loaded with the incremented value of CAR during a call mircoinstruction (BR = 01) Provided that the status bit condition is satisfied (T=1)

2 What ~~a~~ are mapping procedures? Explain

Memory mapping

⇒ Direct mapping In cache memory, each block of main memory maps to only cache lime.

⇒ Associative mapping Any block of main memory can be loaded into any line of the cache it reduces conflicts but requires more complex hardware to chech all cache Lines.

⇒ Set-Associative mapping A Compromise between direct and associative mapping. the cache is divided into sets, and each block maps to any line within a specific set.

Virtual memory mapping

⇒ paging Divides virtual memory into fixed size pages and Physical memory into frames pages are mapped to frames allowing non-contiguous memory allocation and efficent use of physical memory.

⇒ segmentation Divides memory into segments based on Logical division like function or data structure each segment has a base address and limit.

Input/output mapping

⇒ Memory mapped I/O. i/o devices are mapped into the address space of the processor allowing the same instruction used for accessing memory ~~for~~ to be used for I/O operations.

File mapping

⇒ Memory mapped files maps the contents of a file into the virtual memory space of a process. This allows a file to be accessed as if it were part of the program's memory enabling efficent file manipulation.

Address mapping

⇒ logical to physical Address mapping converts logical address generated by the CPU to physical addresses in memory. This mapping is managed by the memory management unit (MMU).

Instruction mapping

⇒ opcode mapping Translate high-level machine instructions into low level operations or micro operation exection executed by the processor's control unit.

3. What about Booth multiplication Algorithm using flow chart and numerical example.

Booth algorithm requires examination of the multiplier bits and shifting of partial product prior to the shifting the multiplicand may be added to the partial product, substracted from the partial or left unchanged according to the following rules.

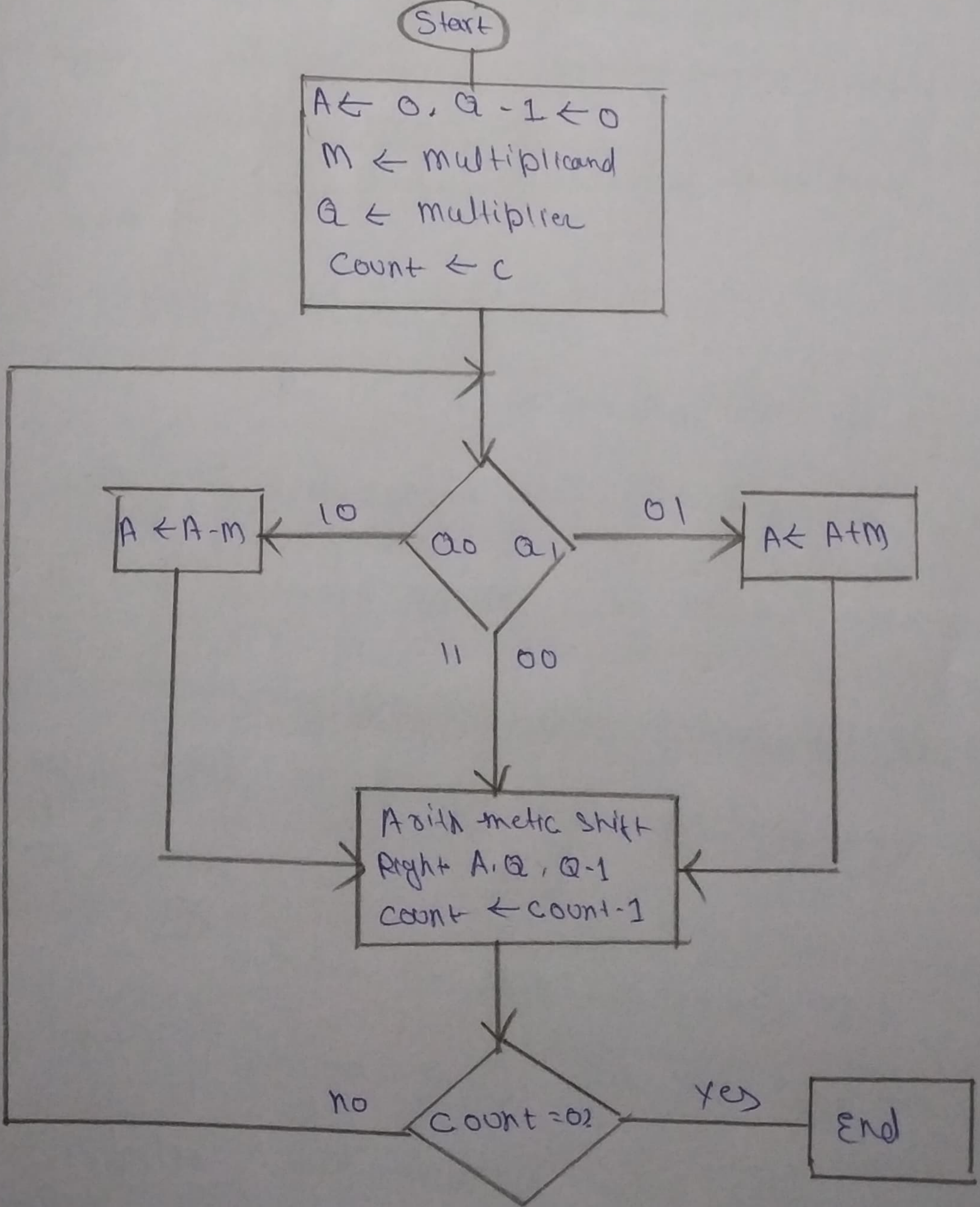
⇒ The multiplicand is subtracted from partial product upon encountering the first least significant 1 in a string of 1's in the multipler.

⇒ The multiplicand is added to the partial product upon encountering the first 0 in a string of 0's in the multiplier.

⇒ The partial product does not change when multiplier bit is identical to the previous multipler bit.

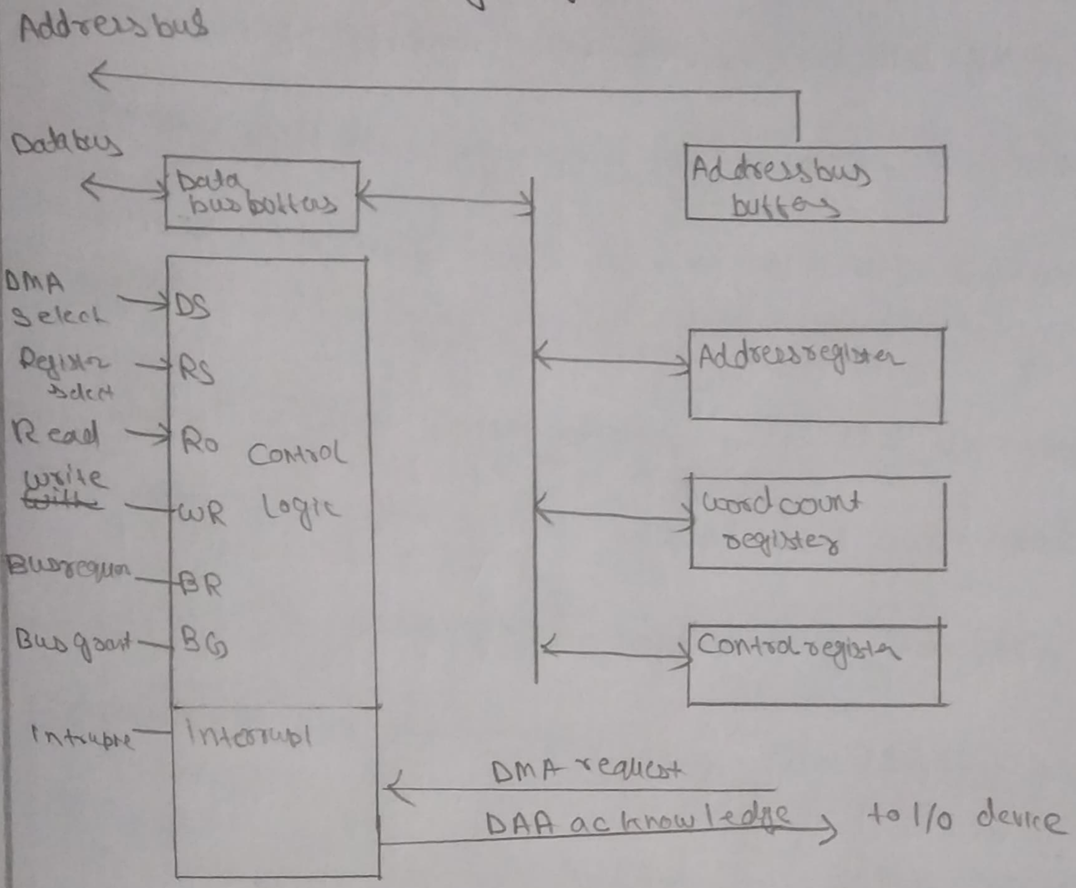
The algorithm works for positive or negative multipliers in 2's complement representtion. This is because a negative multiplier ends with a string of 1's and the last operation will be a subtraction of the appropriate weight.

The two bits of the multiplier in Qn and Qn+1 are inspected. if ~~the~~ the two bits are equal to 10. It means that the first 1 is a string of 1's has been encountered. This requires a subtraction of the multiplicand from the partial product in Ac. If the two bits are equal to 01. it means that the first 0 in a string of 0's has been encountered.



4

Explain the working of DMA Controller



Direct memory Access (DMA)

The unit communicates with the CPU via the data bus and control lines. The registers in the DMA are selected by the CPU through the address bus by enabling the DS (DMA Select) and RS (register select) input the RD (read) and wr (write) input are bidirectional.

when the BG (bus grant) input is 0, the CPU can communicate with the DMA registers throughthe data bus to read from or write to the DMA registers. when BG = 1, the CPU has relinquished (ceased) the buses and the DMA can communicate directly with the memory by specifying an address bus and activating the RD or WR control.

The DMA communicates with the external peripheral through the requst and acknowledge lines by using a prescribed handshaking procedure. The DMA controller has three registers. an address register a word count register. and a control register. The address register contains an address to specify the desired location menory. All registers in the DMA appear to the CPU as I/O interface registers Thus the CPU can read from or write into the DMA registers under program contool via the data bus.

5. What is pipelining ? Explain about Arithmetic pipelining.

Pipelining is the process of accumulating instruction from the processor through a pipeline. It allow storing and executing instruction in an orderly process it is also know as pipeline

Arithmetic pipelining.

Arithmetic pipelines are usually found in most of the computers. They are used for floating point operations. multiplication of fixed point numbers etc for examble the input to the floating point Adder pipelines is

X = A* ^a

Y = B*2^b

Here A and B are mantissas (significant degit of floating point number) while, a and b are exponends

The floating point addition and subtraction

Compare the exponents

Align the mantissas

·Add or substract mantissas

produce the result.