EXPERIMENT NO. 03

DATE OF PERFORMANCE:	**GRADE:**
DATE OF ASSESSMENT:	**SIGNATURE OF LECTURER/ TTA:**

AIM: Implementation Of Arithmetic Microoperations and Logic Microoperations.

THEORY:

In computer central processing units, micro-operations are detailed low-level instructions used in some designs to implement complex machine instructions. Usually, micro-operations perform basic operations on data stored in one or more registers. It also transfers data between registers or between registers and external buses of the central processing unit, and performs arithmetic or logical operations on registers. In a typical fetch decode and execute cycle, each step of a macro-instruction is decomposed during its execution so the CPU determines and steps through a series of micro-operations. The execution of micro-operations is performed under control of the CPU's control unit, which decides on their execution while performing various optimizations such as reordering, fusion and caching.

The basic arithmetic microoperations are addition, subtraction, increment, decrement and shift.

The arithmetic microoperation defined by the statement:

R3<-R1+R2

Table shows the some Arithmetic Microoperations:

Example	Description
R3 ← R1 + R2	Addition contents of R1 plus R2 transferred to R3
R3 ← R1 - R2 , R3← R1 + R2' + 1	Subtraction contents of R1 minus R2 transferred to R3
R2 ← R2'	Complement (really a logic operation)
R2 ← R2' + 1)	Negation
R1 ← R1 + 1	Increment the content of R1 by one
R1 ← R1 - 1	Decrement Increment the content of R1 by one

Binary adder:

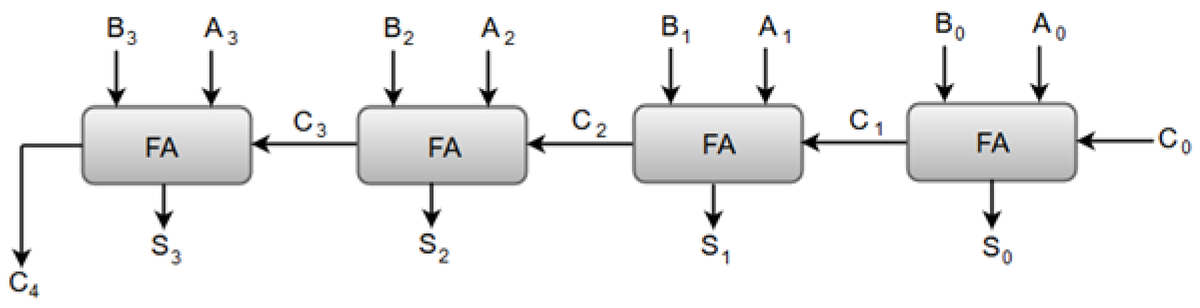
The Add micro-operation requires registers that can hold the data and the digital components that can perform the arithmetic addition.

A Binary Adder is a digital circuit that performs the arithmetic sum of two binary numbers provided with any length.

A Binary Adder is constructed using full-adder circuits connected in series, with the output carry from one full-adder connected to the input carry of the next full-adder.

The following block diagram shows the interconnections of four full-adder circuits to provide a 4-bit binary adder.

4 bit binary adder:



The augend bits (A) and the addend bits (B) are designated by subscript numbers from right to left, with subscript '0' denoting the low-order bit.

The carry inputs starts from C0 to C3 connected in a chain through the full-adders. C4 is the resultant output carry generated by the last full-adder circuit.

The output carry from each full-adder is connected to the input carry of the next-high-order full-adder.

The sum outputs (S0 to S3) generates the required arithmetic sum of augend and addend bits.

The *n* data bits for the A and B inputs come from different source registers. For instance, data bits for A input comes from source register R1 and data bits for B input comes from source register R2.

The arithmetic sum of the data inputs of A and B can be transferred to a third register or to one of the source registers (R1 or R2).

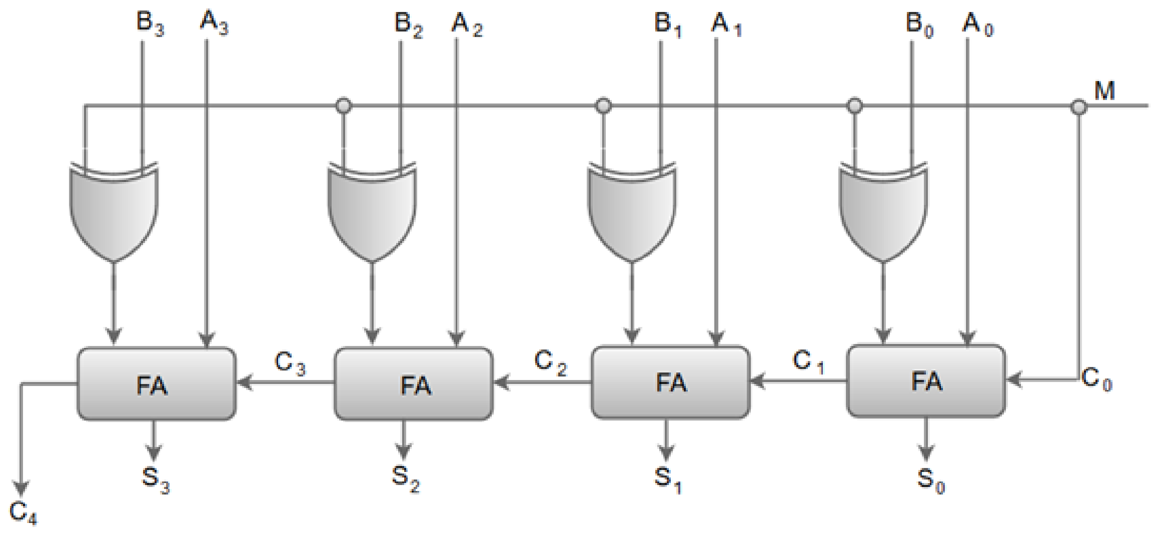
Binary-Adder Subtractor:

The Subtraction micro-operation can be done easily by taking the 2's compliment of addend bits and adding it to the augend bits.

The Arithmetic micro-operations like addition and subtraction can be combined into one common circuit by including an exclusive-OR gate with each full adder.

The block diagram for a 4-bit adder-subtractor circuit can be represented as:

4 bit adder-subtractor:



When the mode input (M) is at a low logic, i.e. '0', the circuit act as an adder and when the mode input is at a high logic, i.e. '1', the circuit act as a subtractor.

The exclusive-OR gate connected in series receives input M and one of the inputs B.

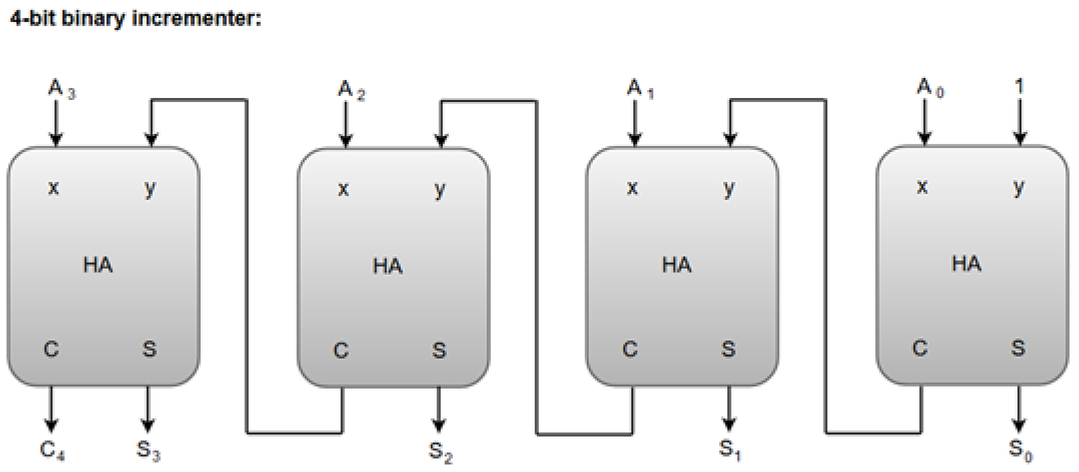
When M is at a low logic, we have B⊕ 0 = B.
The full-adders receive the value of B, the input carry is 0, and the circuit performs A plus B.

When M is at a high logic, we have B⊕ 1 = B' and C0 = 1.
The B inputs are complemented, and a 1 is added through the input carry. The circuit performs the operation A plus the 2's complement of B.

Binary Incrementer

The increment micro-operation adds one binary value to the value of binary variables stored in a register. For instance, a 4-bit register has a binary value 0110, when incremented by one the value becomes 0111.

The increment micro-operation is best implemented by a 4-bit combinational circuit incrementer. A 4-bit combinational circuit incrementer can be represented by the following block diagram.



A logic-1 is applied to one of the inputs of least significant half-adder, and the other input is connected to the least significant bit of the number to be incremented.

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

The binary incrementer circuit receives the four bits from A0 through A3, adds one to it, and generates the incremented output in S0 through S3.

The output carry C4 will be 1 only after incrementing binary 1111.

Logic Microoperations:

Logic Microoperations specify binary operations performed for strings of bits in registers.

These operations consider each bit of the register separately and treat them as binary variables.

Example

$R3 \leftarrow R1 \oplus R2$

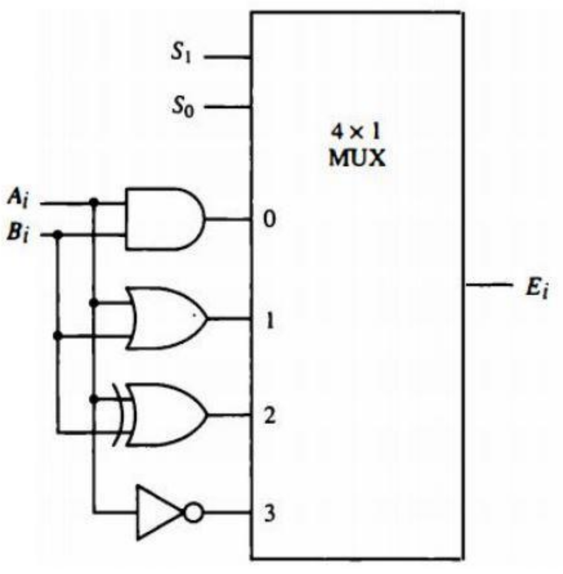
R1 1 0 1 0

R2 $\oplus$ 1 1 0 0

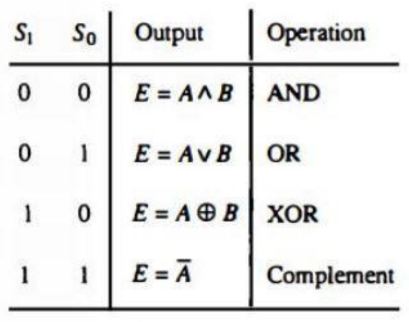
R1 after P=1.   0 1 1 0

Where P is a control function

Logic diagram of Hardware Implementation of Logic Circuit



(a) Logic diagram



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

(b) Function table

Here only one stage is defined for 1 bit. If there are n number of bits then n such stages has to be defined. Applications of Logic Microoperations

1. **Selective Set Operation - It sets 1 to the bits in register A where there are corresponding 1's in register B.It does not affect bit positions that have 0's in B.**
   The OR microoperation can be used to selectively set bits of a register.
   Example:
   consider register A contains 1010 and register B contains 1100. The bits in A corresponding to the bit 1 in the register B will be changed to 1. Therefore, bits 1 and 0 in the register A corresponding to the bits 1 and 1 in the register B will be changed to 1 and 1.
   1 0 1 0. A before
   1 1 0 0 B(logic operand)
   1 1 1 0 A after
2. **Selective Complement Operation-**

The selective Complement operation complements bits in A where there are corresponding 1's in B.

It does not affect bit positions that have 0's in B.

The exclusive OR microoperation can be used to selectively set bits of a register.

Example:

1 0 1 0. A before

1 1 0 0 B(logic operand)

0 1 1 0 A after

3. Selective Clear Operation-

The selective clear operation clears to 0 the bits in A where there are corresponding 1's in B.

It does not affect bit positions that have 0's in B.

The selective clear operation can be achieved by the microoperation A ← A∧B'

Example:

1 0 1 0. A before

1 1 0 0 B(logic operand)

0 0 1 0 A after

4. Mask operation - It is similar to selective clear operation except that the bits of A are cleared only where there are corresponding 0's in B.

The mask operation is an AND Micro Operation.

Example:

1 0 1 0. A before

1 1 0 0 B (logic operand)

1 0 0 0 A after

5. Insert operation - The insert operation inserts a new value into a group of bits.

It incorporates two operations that are masking and then ORing them with required value. The mask operation is an AND microoperation and the insert operation is an OR microoperation.

Example:

Suppose the binary number 1001 is to be inserted in place of 0110

Let the register A contain the bit 0110 1010.

0 1 1 0 is replaced by 0 0 0 0 and remaining bits of A will be 1 1 1 1

Mask

A 0 1 1 0 1 0 1 0

B. 0 0 0 0 1 1 1 1. (In register B the position of bits, wherever the new bits are to be inserted in the register A are filled with zeros and remaining bits are 1

After performing AND operation between the contents of the registers A and B, the output will be as follows Output A 0 0 0 0 1 01 0. AND microoperation

Now, the new bits to be inserted in register A are placed in register B at the masked bit position and remaining bits of register B are zero. Then, OR operation is performed between the contents of register A and register B.

A 0 0 0 0 1 0 1 0

B. 1 0 0 1 0 0 0 0. OR microoperation

1 0 0 1 1 0 1 0

6. Clear operation - The clear operation compares the words in A and B and produces an all 0's result if the two numbers are equal. It is achieved by Exclusive OR operation.

Example:

1 0 1 0 A

1 0 1 0 B

0 0 0 0