(ii) Arithmetic micro-operation :-

These micro-operation perform simple arithmetic operations on numeric data stored in registers. The basic arithmetic micro-operations are addition, subtraction, Increment, decrement and shift. Addition micro-operation is specified as:

$$R_3 \leftarrow R_1 + R_2$$

It means that the contents of register R_1 are added to the contents of register R_2 and the sum is transferred to register R_3. This operation requires three registers to hold data along with the binary adder circuit in the ALU. Binary adder is a digital circuit that generates the arithmetic sum of two binary numbers. Add micro-operation, in a accumulator machine, can be performed as:

$$AC \leftarrow AC + DR$$

Subtraction is most often implemented in machines through complement and add operation. It is specified as:

$$R_3 \leftarrow R_1 - R_2$$

$$R_3 \leftarrow R_1 + (2\text{'s complement of } R_2)$$

$$R_3 \leftarrow R_1 + (1\text{'s complement of } R_2 + 1)$$

$$R_3 \leftarrow R_1 + \overline{R_2} + 1$$

where $\overline{R_2}$ implies 1's complement of R which is bitwise complement.

The increment micro-operation adds 1 to a number in a register. This operation is designated as:

$$R_1 \leftarrow R_1 + 1$$

This can be implemented in hardware by using a binary-up counter.

The decrement micro-operation subtracts 1 from a number in a register. This operation is designated as:

$$R_1 \leftarrow R_1 - 1$$

This can be implemented using binary-down counter.

(iii) Logic Micro-operations :-

Logic operations are basically binary operations, which are performed on the string of bits stored in the registers. For a logic micro-operation, each bit of a register is treated as a variable. A logic microoperation: $R_1 \leftarrow R_1 \cdot R_2$ specifies AND operation to be performed on the contents of R_1 and R_2 and store the results in R_1. For example, If R_1 and R_2 are 8-bit registers and:

R_1 contains 1001 0011

& R_2 Contains 0101 0101

Then R_1 will contain 0001 0001 after AND operation.

Some of the common Logic micro-operations are AND, OR, NOT or complement, Exclusive OR, NOR and NAND.

List of Logic Micro-operations :-

There are 16 different Logic operations that can be performed with two binary variables. They can be determined from all possible truth tables obtained with two binary variables as shown in table.

	Truth Table for 16 Functions of Two variables															
x y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

The 16 Boolean functions of two variables x and y are expressed in algebraic form in the first column of the table shown below. The 16 logic micro-operations are derived from these functions by replacing the variable x by the binary content of register A and variable y by binary content of register B. The logic micro-operations listed in the second column represent a relationship between the binary content of two registers A & B.

Table: 16 Logic Micro-operations

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

Logic micro-operations are very useful for manipulating individual bits or a portion of a word stored in a register. They can be used to change bit values, delete a group of bits or insert new bits values into a register. Examples are

* Selective set :- The selective operation sets to 1 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 following numerical example clarifies this operation:

```
1010   A (before)
1100   B (logic operand)
----
1110   A (after)
```

The OR micro-operation is used for this purpose.

* Selective complement :- 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. For example,

```
1010   A (before)
1100   B (logic operand)
----
0110   A (after)
```

The exclusive-OR micro-operation can be used to selectively complement bits of a register.

* Selective clear :- The selective clear operation clears to 0 the bits in A only where there are corresponding 1's in B. For example:

```
1010    A (before)
1100    B (Logic operand)
----
0010    A (after)
```

The corresponding logic micro-operation is

$$A \leftarrow A \wedge \bar{B}$$

* **MASK** :- The mask operation is similar to the selective-clear operation except that the bits of A are cleared only where there are corresponding 0's in B. For example,

```
0110 1010    A (before)
0000 1111    B
---------
0000 1010    A (after
```

The AND micro operation is used for this purpose.

* **Insert** :- The insert operation inserts a new value into a group of bits. This is done by first masking the bits and then ORing them with the required value.

For example, Suppose that register A contains eight bits 0110 1010. To replace four left-most bits by the value ~~1010~~ 1001, we first mask four unwanted bits:

```
0110 1010    A (before)
0000 1111    B (mask)
---------
0000 1010    A (after masking)
```

and then insert the new value

```
0000 1010    A (before)
1001 0000    B (Insert)
---------
1001 1010    A (after insertion)
```

AND micro-operation is used for mask and OR micro-operation is used for insert operation.